

VU Research Portal

Proceedings of the Seventy-ninth European Study Group Mathematics with Industry

Planque, R.; Bhulai, S.; Hulshof, J.; Kager, W.; Rot, T.O.

2012

[Link to publication in VU Research Portal](#)

citation for published version (APA)

Planque, R., Bhulai, S., Hulshof, J., Kager, W., & Rot, T. O. (2012). *Proceedings of the Seventy-ninth European Study Group Mathematics with Industry*. VU Uitgeverij.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

E-mail address:

vuresearchportal.ub@vu.nl

Proceedings of the 79th European Study Group Mathematics with Industry

VU University Amsterdam, The Netherlands, 24-28 January, 2011

Editors:

Bob Planqué
Sandjai Bhulai
Joost Hulshof
Wouter Kager
Thomas Rot

The European Study Group Mathematics with Industry 2011 was financially supported by:

Technology foundation STW
The Netherlands Organisation for Scientific Research (NWO)
VU University Amsterdam
European Consortium for Mathematics in Industry (ECMI)
Research Training Network Programme Front-singularities (Cordis)

ISBN: 9789086596256

Preface

These are the proceedings of the 79th European Study Group with Industry, held at the VU University in Amsterdam, in the last week of January 2011. These studygroups originate in the tradition of British applied math. For some information on the history of SWIs, see <http://miis.maths.ox.ac.uk/>. In the Netherlands the Studygroups are an annual event. The companies participating in the 2011 studygroup were Deltares, Chess, Statkraft, MARIN, NXP and the Netherlands Forensic Institute. This booklet contains 6 articles in Dutch, written for a more general audience by Bennie Mols, and 6 mathematical chapters which constitute the scientific proceedings. Some problems were actually solved within the week, some were solved later, and some were just put in a different perspective.

Looking back, the participating companies valued the outcome from different perspectives, as is best appreciated from the Dutch articles. I am inclined to group the results as follows. Deltares saw their engineering approach tested and confirmed. Statkraft had to draw the conclusion that there is no simple answer to the question they posed. The NFI received alternatives they perhaps are not yet free to accept, but that will hopefully have a future impact on current juridicial practice, as the use of priors is likely to remain controversial. Nothing to boast on? That is for the reader to judge. As for the other three companies: Chess is using two new algorithms developed during the week, and contemplating a third. MARIN (which participates every year) were impressed how the effective combination of open source mathematical software and the implicit function theorem lead to unexpected insights. Finally NXP had not anticipated the prototype for automatic chip design developed during the week. NXP's designers have taken this initial product on board and are extending it to suit their needs.

SWI2011 would not have been possible without our financial and moral supporters. This allowed all participants to learn a lot, enriched by new horizons, in and outside mathematics. For the organisers it was a stimulating experience to talk to all the companies, also the ones that could not make it to the 2011 program, and to learn from each others skills and knowledge. Finally I would like to thank my coorganisers, in particular Thomas Rot, for the active lead they took, before, during and after the event.

Joost Hulshof

Contents

Nederlandse Samenvattingen	7
1 Wiskunde die overstromingen opvangt	8
2 Energiemanagement van een draadloos sensornetwerk	13
3 Veiling in reservestroom	18
4 Scheepsmanoeuvres moeten tegen een stootje kunnen	22
5 Kortste routes op een analoge chip	26
6 Was de doodrijder ook een hardrijder?	30
Optimal Flood Control	35
<i>Peter Dickinson, Joost Hulshof, André Ran, Majid Salmani and Martijn Zaal</i>	
Node counting in wireless ad-hoc networks	49
<i>Joep Evers, Demeter Kiss, Wojtek Kowalczyk, Tejaswi Navilarekallu, Michiel Renger, Lorenzo Sella, Vincent Timperio, Adrian Viorel, Sandra van Wijk, Albert-Jan Yzelman</i>	
Strategic bidding in a primary reserve auction	75
<i>Denes Palvolgyi, Gergely Csapo, Meike Wortel, Thijs Ruijgrok, Wander Wadman, Zsombor Meder</i>	
Analysis of a Model for Ship Maneuvering	83
<i>M. Apri, N. Banagaay, J.B. van den Berg, R. Brussee, D. Bourne, T. Fatima, F. Irzal, J. Rademacher, B. Rink, F. Veerman and S. Verpoort</i>	
Routing for analog chip designs at NXP Semiconductors	117
<i>Marjan van den Akker, Theo Beelen, Rob H. Bisseling, Bas Fagginger Auer, Frederik von Heumann, Tobias Müller and Joost Rommes</i>	
Statistical modelling of pre-impact velocities in car crashes	133
<i>Wouter Kager, Ivan Kryven, Keith W. Myerscough, Timo van Opstal, Thomas Rot</i>	

Nederlandse Samenvattingen

1 Wiskunde die overstromingen opvangt

Bennie Mols

Eén van de doelen van het Nederlandse bedrijf Deltares is om de waterhuishouding van het landelijke rivierensysteem zo goed mogelijk te controleren. Deltares wil graag weten of de ingenieursaanpak die het zelf heeft ontwikkeld verbeterd kan worden door de wiskundige achtergrond beter te begrijpen.

Nederland Waterland ligt vol met dijken, dammen, kanalen, uiterwaarden, sluizen, pompen en gemalen die zorgen dat polders niet onder lopen, het wassende rivierwater binnen de perken blijft en het land droge voeten houdt. Het Nederlandse instituut Deltares probeert fundamenteel onderzoek op het terrein van waterbeheer van rivierdelta's, kustregio's en riviergebieden te vertalen naar de praktijk. Deltares staat voor de uitdaging om het Nederlandse waterbeheer slimmer te sturen, om energiekosten te besparen en de veiligheid te verbeteren. Daarnaast moet idealiter ook de wisselwerking tussen verschillende waterwerken worden meegenomen, iets wat tot nu toe niet gebeurt maar in de nabije toekomst wel gaat veranderen. Deltares heeft zo'n achthonderd mensen in dienst, gevestigd in Delft en Utrecht. Het bedrijf ontstond in 2008 uit een fusie van GeoDelft, Delft Hydraulics en delen van TNO en Rijkswaterstaat.

Veel sturing van de Nederlandse waterwerken gebeurt nog handmatig. Gaat de waterstand lokaal ergens omhoog, dan kan iemand een pomp aanzetten. "Momenteel worden alleen de huidige waterstanden bekeken", vertelt senioronderzoeker Dirk Schwanenberg van Deltares. "In de toekomst willen we ook de weersvoorspelling meenemen in de sturing. Als je weet dat er over een paar uur veel regen gaat vallen, dan kun je een pomp eerder aanzetten, om het waterniveau al wat omlaag te brengen en ruimte te maken voor het regenwater dat nog gaat komen. We zoeken nu naar manieren om hoe we de weersvoorspelling het beste kunnen meenemen."

In 2011 voerde Deltares al twee pilotprojecten uit die rekening houden met de weersvoorspelling: een project voor het waterschap Dommel en het waterschap Aa en Maas, en een tweede project voor het waterschap Noorderzijlvest in Groningen, Friesland en Drente.

"Een bepaald watersysteem wordt beschreven met wiskundige vergelijkingen die vertellen hoe het watersysteem zich gedraagt", zegt Schwanenberg over de wiskundige achtergrond van het sturingsprobleem. "Dat is een stelsel niet-lineaire gewone differentiaalvergelijkingen. Eén belangrijke randvoorwaarde wordt bepaald door de weersvoorspelling. De weersvoorspelling geeft aan hoe veel water je op een bepaald moment verwacht. De tweede randvoorwaarde wordt bepaald door de manieren waarmee we de waterafvoer kunnen beïnvloeden, bijvoorbeeld met pompen of sluizen. Nu kijken wij bij Deltares als ingenieurs naar het sturingsprobleem. Wij weten dat bepaalde dingen werken, maar we weten niet wat er wiskundig precies achter zit en daar wilden we achter komen. Onze vraag aan de studiegroep Wiskunde met de Industrie was dan ook of we de sturing beter kunnen optimaliseren wanneer we de achterliggende wiskunde beter begrijpen."



Lagrangemethode

“Het heeft ons veel tijd gekost om de wiskundige achtergrond van het probleem precies in kaart te brengen”, vertelt wiskundige en promovendus Martijn Zaal namens de studiegroep die het Deltaresprobleem eind januari 2011 bestudeerde. “Deltares gebruikt een intuïtieve methode om de optimale strategie te vinden. Hun methode begint met een bepaalde gok voor de oplossing van het optimalisatieprobleem en probeert de oplossing vervolgens stap voor stap te verbeteren. De manier waarop deze verbeteringen werken begrepen we eerst niet.”

Het concrete probleem dat de studiegroep heeft bekeken, bestaat uit vier gekoppelde waterreservoirs, waarbij de waterstroom naar het eerste reservoir op een redelijke tijdschaal voorspelbaar is. Dit probleem staat model voor vier reservoirs in de Rijn-Maasdelta. “Ons idee was nu om hun optimalisatieprobleem te formuleren als een Lagrange-multiplier-probleem. De Lagrangemethode werkt goed bij optimalisatieproblemen waarbij de randvoorwaarden gelijkheden zijn, bijvoorbeeld wanneer er bepaalde behoudswetten gelden. Dingen die je niet kunt kiezen, vat je in deze methode op als dingen die je wel kunt kiezen, maar dan moeten de wiskundige vergelijkingen wel voldoen aan een paar extra voorwaarden.”

Deze methode geeft in het algemeen een groter stelsel vergelijkingen dan het oorspronkelijk geformuleerde stelsel. Maar het voordeel is dat de oplossing met de Lagrangemethode wel optimaliteit garandeert. Met de Lagrange-multiplier-methode kwam de studiegroep uit op een stelsel van zes vergelijkingen. Zaal: “De volgende



vraag is of de volgorde waarin je de vergelijkingen gaat oplossen uitmaakt voor hoe gemakkelijk de oplossing te berekenen is. Voor ons was het interessant om te zien dat de Deltares-methode overeen bleek te komen met een van de oplossingsmethoden voor het Lagrangeprobleem dat wij hebben bekeken.”

De methode van de Deltares-onderzoekers gebruikt een slim algoritme om een bepaalde afgeleide uit te rekenen en die gebruiken ze om de volgende gok in hun strategie te verbeteren. Dat slimme algoritme bleek veel te maken te hebben Lagrange-multipliers en dat dwarsverband verraste de wiskundigen van de studiegroep. Zaal: “Uit een vergelijking van hun oplossingsmethode met onze oplossingsmethode zijn wij tot de conclusie gekomen dat hun strategie goed werkt. Aan de ene kant is het misschien teleurstellend voor hen dat het niet beter lijkt te kunnen, maar aan de andere kant kunnen ze er een bevestiging in zien van de kwaliteit van hun eigen methode.”

Tijdshorizon

De wiskundigen hebben vervolgens de Lagrange-multipliermethode geïmplementeerd voor het concrete probleem van de vier gekoppelde waterreservoirs. “Die berekeningen laten zien dat het heel belangrijk is wat de tijdshorizon van de weersvoorspelling is”, vertelt Zaal. “Wanneer je de weersvoorspelling voor de komende tien uur goed kent, dan kan dat tot een andere strategie leiden dan wanneer je alleen maar de weersvoorspelling voor de komende twee uur kent. Hoe korter de tijdshorizon, hoe minder mogelijkheden je hebt om te anticiperen.”

Weersvoorspellingen worden natuurlijk voortdurend geactualiseerd. Als dan telkens helemaal opnieuw moet worden berekend wat er met de waterstromen gebeurt, is



dat meestal inefficiënt. De weersvoorspelling van nu hangt namelijk sterk samen met die van over een paar uur. De vraag is dan wat een efficiënte manier is om rekening te houden met de weersvoorspelling zonder dat je telkens tachtig tot negentig procent van het gereken eerst weggooit en later weer opnieuw gaat doen. Zaal: “Wij zijn tot de conclusie gekomen dat het efficiënter is om de tijdshorizon langer te houden dan vier uur, en de resultaten van de vorige berekening deels te hergebruiken om met een betere schatting te komen.”

De studiegroep heeft het optimalisatieprobleem ook nog met een tweede methode aangepakt. Het stelsel vergelijkingen kan ook numeriek worden opgelost met Niet-Lineair Programmeren (NLP). “Je kunt NLP gebruiken om een beter inzicht in de gevoeligheid voor parameters te krijgen”, zegt Zaal. “En dat is weer handig om inzicht te krijgen in welke invloed een verandering van de weersvoorspelling heeft op de oplossing van het optimalisatieprobleem. Met deze methode hebben we laten zien hoe je de oplossing van het optimalisatieprobleem in theorie kunt laten meeveranderen met een verandering van de weersvoorspelling. Maar dit is niet meer dan een aanzet van hoe het in de praktijk kan. We hebben niet de tijd gehad om deze aanpak in een computerprogramma te implementeren.”

Kennisexport

“Wij kunnen er geen beter product mee maken”, zegt Dirk Schwanenberg van Deltares over de resultaten van de studiegroep, “maar hun werk heeft ons wel een beter begrip gegeven van de wiskundige achtergrond van het probleem. Ik heb het heel nuttig gevonden om te zien hoe mensen met wiskundige achtergrond aankijken tegen

een typisch ingenieursprobleem als het onze. Wij hadden ook niet direct een grote doorbraak verwacht. Eigenlijk heeft de studiegroep aangetoond dat wij het probleem al vrij goed oplossen. De studiegroep heeft ook laten zien dat je onder bepaalde omstandigheden het sturingsprobleem sneller en slimmer kunt oplossen. Nadeel is wel dat de oplossingsmethode dan minder generiek wordt. Dat is voor ons geen optie.”

Deltares is inmiddels begonnen om het meenemen van de weersvoorspelling te implementeren in hun producten. Schwanenberg: “Dit jaar zijn we begonnen onze kennis te exporteren naar de Verenigde Staten, Brazilië en Uruguay. Daar gaat het vaak om watersystemen die waterkrachtcentrales bevatten. Dat levert een soort omgekeerd systeem van wat we in Nederland hebben, maar veel componenten zijn identiek aan wat we in Nederland hebben.”

De betere sturing waarop Deltares zich richt maakt ook onderdeel uit van het nationale onderzoeksproject Flood Control 2015, dat onder andere via sensoren in dijken en real-time informatie over weer en waterstanden de wateroverlast beter probeert te voorspellen en te beheersen. “De sturing van het watersysteem biedt nog veel mogelijkheden om het beter te doen”, besluit Schwanenberg.

2 Energiemanagement van een draadloos sensornetwerk

Bennie Mols

Steeds meer toepassingen bevatten draadloze sensoren die in een netwerk met elkaar communiceren. Hoe kunnen die sensoren met zo veel mogelijk burens praten en toch zo min mogelijk energie verbruiken?

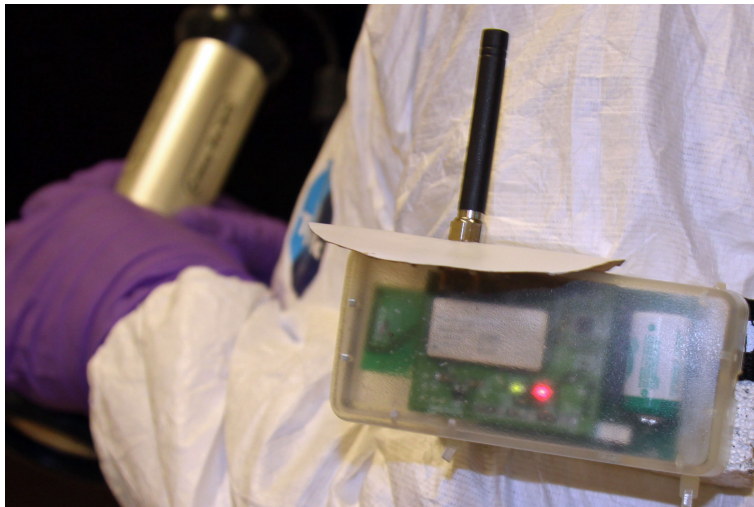
Het is 's avonds laat wanneer je in je auto komt aanrijden bij een tankstation. Buiten springt er één lamp aan. Je rijdt verder en er springt nog een lamp aan en zo gaat het door. Het lijkt wel of je wordt gevolgd. Je voelt je onprettig. Om dat gevoel te voorkomen, wordt er gewerkt aan subtielere verlichting waarbij lampen niet slechts aan of uit zijn, maar op een prettige manier worden gedimd. Een tankstation krijgt zo een slimme, energiezuinige verlichting die ongeveer weet waar jij als automobilist rijdt en waar het dus lichter moet zijn, zonder je het gevoel te geven dat je wordt gevolgd. Voor zo'n slim sensornetwerk is meer elektrische bekabeling nodig. Dat is duur; vandaar dat er wordt gekozen voor draadloze sensoren die met elkaar praten. Het sensornetwerk moet weten waar een automobilist precies rijdt om daar een prettige verlichting op maat te verzorgen.

Het Nederlandse bedrijf Chess maakt draadloze sensoren voor deze tankstation-toepassing, maar ook voor vele andere toepassingen. Chess is gespecialiseerd in de ontwikkeling van hard- en software op maat. Behalve draadloze sensoren heeft het bedrijf bijvoorbeeld een real-time-uitlezing van een MRI-hersenscanner ontwikkeld, een manier om met een laserbundel de doorbuiging van een vliegtuigvleugel te meten, maar ook systemen voor microbetalingen. Chess is gevestigd in Haarlem en heeft ongeveer honderdvijftig medewerkers.

In 2006 deed het bedrijf ook al eens mee aan de Studiegroep Wiskunde met de Industrie. "Dat is ons goed bevallen", zegt Chess-onderzoeker Bert Bos, "vandaar dat we graag weer eens wilden meedoen. In 2006 ging het om de betrouwbaarheid van een draadloos sensornetwerk dat een bosbrand detecteert. Dit keer wilden we graag weten hoe een draadloos sensornetwerk zo goed mogelijk kan communiceren bij een zo laag mogelijk energieverbruik."

Radiocommunicatie

Een enkele draadloze sensor is klein, goedkoop en energiezuinig. Zijn processorkracht en geheugen zijn beperkt. In een netwerk communiceren de sensoren met elkaar op een en dezelfde radiofrequentie. Elke andere sensor die binnen het radiobereik ligt, kan het signaal van de zender ontvangen. Maar hoeveel ontvangende burens dat zijn, hangt sterk af van de omstandigheden, zoals obstakels onderweg of de hoeveelheid vocht in de lucht. Het bereik kan variëren van vijf tot vijftig meter. Het aantal burens dat het signaal van een zender ontvangt kan bovendien van moment tot moment variëren. Verder is het probleem asymmetrisch: dat sensor *B* een bericht van sensor *A* kan ontvangen, betekent niet automatisch dat sensor *A* ook een bericht van sensor *B* kan ontvangen. Wanneer twee of meer sensoren precies tegelijk zenden, dan storen ze elkaar en horen de andere betrokken sensoren niets. Chess staat daarom voor de opdracht om een zo efficiënt mogelijk communicatieprotocol voor een dynamisch netwerk van sensoren te ontwikkelen.



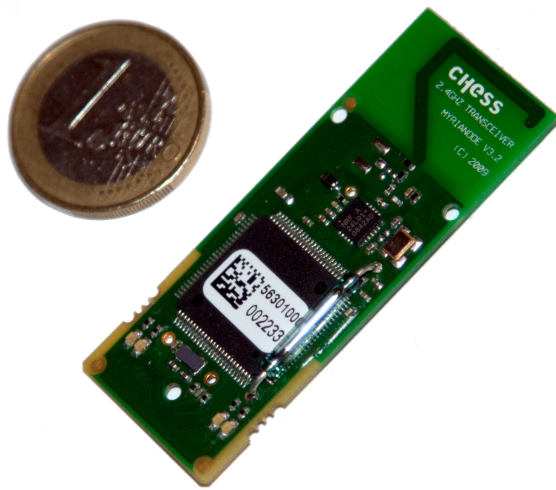
Figuur 2.1: Draadloze sensor van Chess.

Om zo lang mogelijk met hun batterijen te doen, doen de sensoren het gros van de tijd niets en communiceren ze alleen maar in een korte tijdsperiode. Typisch zijn ze in een paar milliseconden actief en doen ze de rest van de seconde niets. Zo is elke sensor lang inactief, kort actief, lang inactief enzovoort. “Bij het ontwerpen van een communicatieprotocol”, zegt Bos, “ontstaat een spanningsveld tussen het aantal burens dat signaal ontvangt van een sensor en zijn energieverbruik. Hoe meer burens, hoe beter in principe de communicatie verloopt, maar ook hoe meer energie het netwerk verbruikt. Wij willen dus graag het optimum aantal burens weten waarmee een sensor moet communiceren.”

Random identificatienummers

Albert-Jan Yzelman was een van de tien wiskundigen die in januari 2011 een week lang aan het probleem heeft gewerkt. “Wij hebben ons geconcentreerd op de beantwoording van twee vragen”, vertelt Yzelman. “Allereerste de vraag: Wat is een schatting voor het aantal burens dat een bericht van een bepaalde sensor ontvangt?” Om die vraag te beantwoorden hebben we vier algoritmen ontwikkeld. De tweede vraag die we hebben onderzocht is Wat is een schatting voor het aantal sensoren dat zich op een bepaald moment in het netwerk bevindt?” In toepassingen waarin de sensoren kunnen bewegen denk aan sensoren op vogels of mensen varieert het aantal sensoren dat met elkaar kan communiceren in de tijd. Sensoren kunnen immers tijdelijk buiten bereik van alle andere sensoren vallen. Voor de beantwoording van deze vraag hebben we zes algoritmen ontwikkeld.”

Sommige algoritmen hadden een statistisch karakter, andere waren deterministisch. Voor elk algoritme schreven de wiskundigen een pseudocode: een beknopte uitwerking van het algoritme in een programma-achtige structuur. Een programmeur kan deze pseudocode gebruiken om in de gewenste programmeertaal het algoritme te implementeren. Elk algoritme werd vervolgens geïmplementeerd en getest in een



Figuur 2.2: Sensor voor lokalisatie bij Crime Scene Investigation The Hague (2011).

simulatie die verschillende netwerkgroottes (van 100 tot 1000 sensoren) en een verschillend aantal communicatiecycli (30 en 300 stappen) gebruikte.

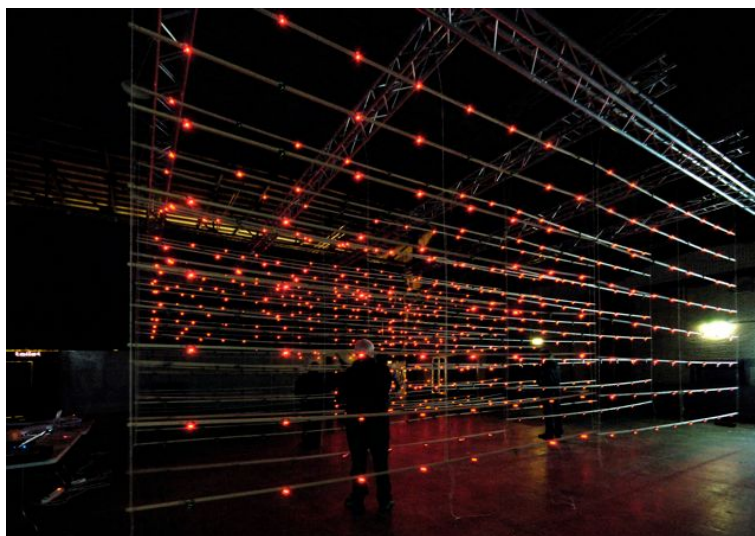
Uit deze simulaties bleek dat als schatter voor het aantal burens het zogeheten Random ID-algoritme het beste presteerde. Yzelman: “Het idee van dit algoritme is dat elke sensor op een willekeurige manier een identificatienummer tussen 0 en L kiest. L is typisch groter dan het aantal burens van een sensor maar kleiner dan het aantal sensoren in het netwerk. L mag niet te groot zijn omdat het een sensor dan te veel energie kost om het nummer naar al zijn burens te sturen. Meerdere sensoren in het netwerk zullen dus hetzelfde identificatienummer krijgen. Bij elk bericht dat een sensor verstuurt wordt het identificatienummer meegestuurd.”

Elke sensor onthoudt een vector met bitlengte L waarvan de componenten 0 of 1 kunnen zijn. In het begin worden alle componenten van deze vector op 0 gezet. De plaats van de component wordt bepaald door het identificatienummer van de verzendende sensor. Een 0 geeft aan dat er nog geen bericht van die sensor is ontvangen; een 1 geeft aan dat er wel al een bericht van die sensor is ontvangen. De ontvangende sensor past deze vector bij elke nieuwe communicatiestap aan.

“Na een bepaald aantal communicatiestappen stabiliseert deze vector”, legt Yzelman uit. “Uit deze gestabiliseerde vector kunnen we dan een schatting van het aantal burens berekenen. Als schatter voor het aantal burens presteert dit algoritme verreweg het beste van de vier die we ontwikkeld en getest hebben.”

MinTopK

Het best presterende algoritme om het aantal sensoren in het netwerk te bepalen, bleek het zogeheten MinTopK-algoritme. Yzelman: “Het idee is simpel en elegant en het blijkt nog te werken ook. Net als bij het Random ID-algoritme kiest elke sensor weer willekeurig een identificatienummer tussen 0 en L dat bij elke communicatiestap



Figuur 2.3: Het 1000-node-experiment: dit experiment toont aan dat je geen 1000 nummers wilt toekennen, of wilt gaan tellen op basis van toegekende nummers.

wordt verzonden. Vervolgens laten we elke sensor de K grootste identificatienummers die hij ooit heeft ontvangen onthouden. K kan vrij klein zijn, bijvoorbeeld een getal tussen tien en honderd. Hoe groot dat getal mag zijn, hangt af van hoeveel geheugenruimte een sensor heeft: hoe groter de geheugenruimte, hoe groter K mag zijn. In plaats van een veel langere lijst met een lengte L te onthouden, hoeft een sensor nu alleen maar een veel kortere lijst van lengte K te onthouden.”

De truc is nu dat het kleinste getal van die K getallen, zeg s , een goede schatting blijkt te geven voor de netwerk grootte. s is het minimum van de top- K waarden, vandaar dat het algoritme de naam MinTopK heeft gekregen. Als het interval $[s, 1]$ —dat dus een lengte $(1 - s)$ heeft— K waarden bevat, dan bevat het hele interval $[0, 1]$ ongeveer $K/(1 - s)$ waarden. Yzelman: “Het quotiënt $K/(1 - s)$ is dan een goede schatter voor het aantal sensoren in het netwerk. Ik denk dat de vraag van Chess is beantwoord wanneer ze het MinTopK-algoritme gebruiken. Het algoritme kan nog wel verbeterd worden, maar ik denk dat het nu al in de praktijk bruikbaar is.”

De wiskundigen pasten het eerder gebruikte Random ID-algoritme een beetje aan om ook een schatting te geven van het aantal sensoren in het netwerk. Yzelman: “Ik denk dat Chess ook dit aangepaste Random ID-algoritme kan gebruiken. Nadeel is dat het niet stabiel is: wanneer je meer stappen gebruikt, wordt de fout groter. Maar misschien valt die stabiliteit nog wel te verbeteren.”

Stap naar de toepassing

“De wiskundigen hebben verrassende oplossingsmethoden gepresenteerd, waar ik zelf niet op was gekomen”, vertelt Chess-onderzoeker Bert Bos over de resultaten van de studiegroep. “Voor mij springen twee inzichten er uit. Allereerst het inzicht dat het handig is om alleen maar tijdelijk een random identiteitsnummer toe te kennen aan de sensoren. Identiteiten van sensoren hoeven dan niet eens uniek te zijn. Deze methode

geeft voor ons een aardige maat voor het optimale aantal burens. De nauwkeurigheid van de uitkomsten is misschien nog niet zo hoog, maar ons gaat het vooral om de orde grootte. En die klopt. Het tweede inzicht dat voor ons nuttig inzicht is, is dat we via een stochastische methode kunnen meten hoe lang een bericht onderweg is.”

Chess zoekt naar eenvoudige, effectieve algoritmen die op een goedkope manier een goede oplossing kunnen berekenen, zonder dat dat automatisch de optimale oplossing is. Het MinTopK-algoritme, dat volgens Yzelman het beste presteert, vindt Bos lastiger te begrijpen. Of hij daar verder mee gaat, weet hij nog niet. Met de andere twee algoritmen gaat hij in ieder geval wel verder: “Het komende jaar gaan we ze verder testen, zowel met simulaties als in echte experimenten met een sensornetwerk. Wij hopen dat we ze daarna kunnen implementeren in onze draadloze sensoren.”

3 Veiling in reservestroom

Bennie Mols

In het Europese elektriciteitsgrid moet de energieopwekking te allen tijde overeenkomen met het energieverbruik. Om in te kunnen spelen op een plotseling toenemende energievraag, verplicht de Duitse regelgever energieaanbieders om voldoende energiereserve op voorraad te hebben. Deze energiereserve wordt op een veiling verhandeld. Kan de speltheorie het biedgedrag op deze veiling voorspellen?

Statkraft is een Noors staatsbedrijf dat de grootste producent van hernieuwbare energie in Europa is. Statkraft beheert onder andere een groot aantal waterkrachtcentrales. Zo'n tien jaar geleden besloot het bedrijf zijn energie ook op het Europese vasteland te verhandelen. Als eerste opende het een kantoor in Amsterdam; een jaar volgde een kantoor in het Duitse Düsseldorf. Energieproducent Statkraft verkoopt zijn energie aan energieleveranciers, die op hun beurt de energie weer verkopen aan de eindgebruikers, zoals particulieren en bedrijven.

Duitsland heeft de operators van het elektriciteitsgrid verplicht om reserve-energie op voorraad te hebben voor het geval bijvoorbeeld een plotselinge black-out optreedt. Die reserve-energie moet binnen tien seconden geleverd kunnen worden. "De reserve-energie wordt verhandeld op een maandelijks veiling", vertelt elektrotechnisch ingenieur Philipp Siemes, hoofd van de afdeling Short term Energy Management van Statkraft in Düsseldorf. "De aanbieders bieden daar x megawatt aan tegen een prijs van y euro per megawatt. De veiling werkt zo dat de capaciteiten met de laagste prijs per eenheid-verhoudingen worden gekocht totdat gridoperators hun verplichte reserve-energie hebben gehaald. De uitkomsten van de veiling zijn publiek toegankelijk."

De verzoeken van de gridoperators worden door volautomatisch door software gegenereerd. Technisch is het een veeleisend probleem om binnen een tiental seconden een grote hoeveelheid energie te kunnen leveren. Niet alle energiecentrales kunnen immers binnen zo snel op een sterk toegenomen energiebehoefte inspelen. Als energieproducent wil Statkraft natuurlijk een zo goed mogelijke prijs krijgen op de energieveiling. De vraag is hoe ze dat voor elkaar krijgen.

Siemes had zelf al behoorlijk wat tijd in dit probleem gestopt. Hij probeerde om de energieveiling te begrijpen vanuit fundamentele economische wetten: Wat is de invloed van de energieprijs? Wat is de invloed van de brandstofprijs? Wat zijn de invloeden van diverse andere economische factoren? Siemes: "Ik vond echter geen fundamentele correlatie tussen deze factoren en de werkelijke veilingprijzen. Vandaar dat ik het probleem wilde aanpakken vanuit speltheoretisch oogpunt."

In de speltheorie wordt immers de strategische interactie tussen de diverse rationeel handelende spelers van een 'spel' geanalyseerd. Omdat Siemes niet vertrouwd was met de ins en outs van speltheorie zou het hem veel te veel tijd kosten om het problemen speltheoretisch te analyseren. Siemes: "Dus dacht ik: laat ik het probleem voorleggen aan een stel slimme wiskundigen. Via een Nederlandse collega had ik gehoord over de Studiegroep Wiskunde met de Industrie, en zo is het gekomen dat ik ons veilingprobleem heb voorgelegd aan een studiegroep van wiskundigen."

Behalve het proberen te achterhalen van de beste veilingprijs voor Statkraft, is er nog een reden om de veilingprijzen goed onder de loep te nemen. Juist omdat er



Figuur 3.1: Het hogere reservoir van de Statkraft energieopslagcentrale in Erzhausen (Duitsland). Deze centrale kan de reserve-energie leveren die op veilingen wordt verhandeld. Wanneer de elektriciteit goedkoop is (bijvoorbeeld 's nachts) gebruikt deze centrale energie door water van het lagere naar het hogere reservoir te pompen. Bij de eerstvolgende goede gelegenheid mag water van het hogere naar het lagere reservoir stromen. Daarbij drijft het turbines aan die elektriciteit opwekken. Bron: Statkraft/Hans Starosta

relatief weinig energieaanbieders op de veiling opereren, zou het kunnen dat er stilzwijgende afspraken bestaan tussen die aanbieders om de prijs kunstmatig hoog te houden. Siemes: “Het zou in principe kunnen dat er subtiele strategische samenwerkingen bestaan tussen aanbieders, zonder dat er overduidelijk sprake is van een kartel. In onze traditionele economische analyse konden we zo’n subtiele samenwerking niet vinden, maar misschien dat een speltheoretische analyse zich beter leent om een subtiele samenwerking bloot te leggen.”

Gehele of gedeeltelijke rationaliteit

Een groep van zes wiskundigen boog zich eind januari 2011 een week lang over het veilingprobleem van Statkraft. Promovendus Gergely Csapo van de afdeling Kwantitatieve Economie van de Universiteit Maastricht was een van hen. Hij vertelt dat de groep drie oplossingswegen heeft bewandeld: allereerst een speltheoretische aanpak gebaseerd op perfecte rationaliteit van de spelers; ten tweede een model gebaseerd op gedeeltelijke rationaliteit van de spelers; en ten derde een econometrisch model dat alleen maar is gebaseerd op een tijdreeksanalyse van de veilingresultaten uit het verleden. In de tijdreeksanalyse wordt er helemaal geen aanname gedaan over de rationaliteit van de spelers.

“In de speltheoretische aanpak waarbij de spelers perfect rationeel zijn, hebben we twee soorten markten bekeken”, vertelt Csapo. “Als de markt groot genoeg is zodat er geen dominante speler kan ontstaan, dan zien we dat de prijs per verkochte eenheid daalt totdat de laatste verkoper op die laatste eenheid geen winst meer maakt. Als de markt zo klein wordt dat één speler dominant kan worden, dan zien we inderdaad dat deze de prijs kan gaan opdrijven.”

In de tweede aanpak simuleerden de wiskundigen tweeduizend biedingsrondes van zeven verschillende spelers met zeven verschillende biedstrategieën. In de eenvoudigste gevallen zijn de biedstrategieën bijvoorbeeld het gemiddelde van de geaccepteerde biedingen van de afgelopen maand, of de hoogste bieding van de afgelopen maand.



Figuur 3.2: Windpark in Noorwegen. Bron: Statkraft/Christian Houge

“Ondanks het feit dat gedeeltelijke rationaliteit realistischer is dan perfecte rationaliteit, bleek deze aanpak niet zoveel op te leveren”, concludeert Csapo. “We zien dat de prijs langzaam convergeert naar de kostprijs om de reserve-energie te maken. Dat is niet realistisch voor de energie-aanbieders want dan verdienen ze niets meer.”

In de derde aanpak hanteerden de wiskundigen de algemene econometrische manier van modelleren, waarbij wordt verondersteld dat er a priori niets bekend is over hoe de data tot stand komen. Uitgangspunt waren de geaccepteerde biedingen (hoeveel energie voor welke prijs) tussen december 2007 en januari 2011: een periode met 25 veilingen. Csapo: “We hebben geprobeerd met een autoregressive moving average model (ARMA) een onderliggende structuur te ontdekken, maar dat is niet gelukt. We hebben geen manier gevonden om de bieding op de eerst volgende veiling goed te voorspellen”

Een paar jaar geleden werd de reserve-energie nog elk half jaar geveild. Dat veranderde in eerste instantie naar een maandelijks veiling, en in het afgelopen jaar zelfs naar een wekelijkse veiling. De wiskundigen hebben ook nog onderzocht of ze theoretisch kunnen verklaren wat er dan in de praktijk gebeurt. Csapo: “Wij hebben aangetoond dat het verkorten van de tijd tussen twee veilingen leidt tot hogere prijzen, en dat is precies het omgekeerde van wat de regelgevers hadden gedacht. Maar het komt wel overeen met wat er in werkelijkheid is gebeurd. Ik denk dat dit het belangrijkste resultaat is van ons werk.”

Conclusie

“Zelfs de speltheorie blijkt het veilingprobleem niet te kunnen oplossen”, zegt Philipp Siemes van Statkraft over de resultaten van de studiegroep. “Het lijkt erop dat er geen ultiem model is dat de toekomstige veilingprijzen kan voorspellen. Aan de ene kant

is dat teleurstellend. Maar aan de andere kant is het voor ons als bedrijf aardig om te zien dat een groep slimme wiskundigen niet in staat is geweest om het beter te doen dan wij het hebben gedaan. In ieder geval niet binnen een week. Statkraft heeft het goed gedaan op de veiling.”

Wat is dan Statkrafts strategie op de veiling? Siemes: “Wat wij doen is dat we twee of drie mensen laten kijken naar de trends uit het verleden. Zij bediscussiëren die trends en komen op een vrij subjectieve manier tot een consensus over een nieuw bod. Er ligt geen model aan ten grondslag dat precies berekent hoe wij op de veiling moeten handelen. Uit de resultaten van de wiskundigen blijkt echter ook dat de andere energieproducenten geen model hebben waarmee ze de toekomstige veilingprijzen voorspellen. Als ze zo’n model wel hadden gehad, dan hadden we het in de analyses wel gevonden.”

De speltheoretische aanpak kon wel een praktijksituatie uit 2008 verklaren, vertelt Siemes. “Toen was één producent de hoofdleverancier. Hij kon de prijs opdrijven. De situatie veranderde in 2009, waardoor de prijs naar beneden liep. En dat laat het eerste model van de wiskundigen ook zien. Dat model is voor ons het interessantste. Het model met de gedeeltelijke rationaliteit kon de resultaten uit het verleden niet verklaren en ook de aanpak met de tijdreeksanalyse gaf geen goede fit met de data.”

Siemes weet nog niet hij verder gaat met de speltheoretische aanpak. “Het punt is dat de markt is veranderd. Tijdens de studieweek gebeurde de veiling nog maandelijks. Niet lang daarna werd het een wekelijkse veiling, wat de situatie weer heeft veranderd. Een tweede punt is dat tot nu toe alleen de geaccepteerde biedingen werden gepubliceerd en niet de afgewezen biedingen, die te hoog waren. De Duitse regelgever denkt erover na om te verplichten ook de afgewezen biedingen openbaar te maken. Als dat gaat gebeuren, dan willen we dit nieuwe type veiling zeker opnieuw gaan analyseren. Maar als dat niet gebeurt, dan denk ik dat we bij onze huidige veilingstrategie blijven.”

4 Scheepsmanoeuvres moeten tegen een stootje kunnen

Bennie Mols

Stabiel in een rechte lijn varen is wel het minste wat een schip moet doen. Door de vele parameters die in een scheepsontwerp zitten, is dat toch niet altijd het geval. Maritiem onderzoeksinstituut MARIN zoekt een handige wiskundige manier om die stabiliteit snel te bepalen.

Of het nu gaat om passagiersschepen, containerschepen of marineschepen, een van de meest basale vragen is hoe stabiel het schip blijft wanneer het in rustig water vaart. Zwabbert het bij de minste of geringste verstoring op en neer of blijft het netjes in een rechte lijn varen? Hoe reageert het op een verandering van de roerhoek? En wanneer het een bocht met een bepaalde draaicirkel maakt, gaat het dan wel of niet van die draaicirkel afwijken bij het minste of geringste golfje? Dit is het soort stabiliteitsvragen dat het Nederlandse maritieme onderzoeksinstituut MARIN in Wageningen in opdracht van scheepsontwerpers onderzoekt.

Het MARIN onderzoekt de scheepsstabiliteit in rustig vaarwater enerzijds in realistische experimenten met schaalmodellen, maar anderzijds ook theoretisch. De theoretische aanpak gaat uit van drie bewegingsvergelijkingen van het schip, eentje voor de beweging rond de lengteas, eentje voor beweging rond de as loodrecht op het schip en eentje voor de beweging rond de as dwars op het schip. Wiskundig gezien gaat het om een stelsel gekoppelde tweede orde differentiaalvergelijkingen met veel niet-lineaire termen, wat de oplossing lastig maakt. De coëfficiënten (en ook sommige machten van de niet-lineariteiten) die er in voorkomen worden gemeten in experimenten met schaalmodellen. Een schip kent enkele tientallen belangrijke parameters die in het ontwerp kunnen worden veranderd, van de positie en de grootte van het roer tot de weerstand in het water en de ligging van het massazwaartepunt. Al deze parameters samen bepalen hoe goed een schip in rustig water manoeuvreert.

Traditioneel simuleert het MARIN het scheepsgedrag door het stelsel vergelijkingen numeriek tijdstapje voor tijdstapje op te lossen. Maar om de stabiliteit te bestuderen, moeten de parameters telkens een beetje worden veranderd om weer een nieuwe simulatie uit te voeren. Zo zijn al snel tientallen simulaties nodig, wat veel tijd kost. Dat moet handiger kunnen, dacht Ed van Dalen, senior onderzoeker bij het MARIN en zelf ook opgeleid als wiskundige. “Een fysicus zoekt een stationaire oplossing van tijdafhankelijke vergelijkingen. Voor een wiskundige is die stationaire oplossing een evenwicht in de fase-ruimte. Ik had het gevoel dat er een handigere wiskundige manier moest zijn om de scheepsstabiliteit te onderzoeken dan door telkens met iets andere parameters weer een nieuwe simulatie te draaien.”

De vraag aan de wiskundigen van de Studiegroep Wiskunde met de Industrie was dan ook of zij stabiele evenwichten konden aantonen zonder het scheepsgedrag in de tijd te simuleren.

Voortstuwingsmodel

“Toen wij het stelsel vergelijkingen zagen, dachten we meteen: daar kunnen we wat mee”, vertelt wiskundige en promovendus Frits Veerman van de Universiteit Leiden. “De vergelijkingen waren echter opgesteld door een fysicus. Soms wil een fysicus een

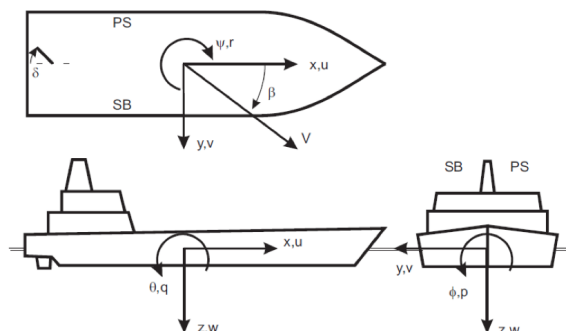


bepaalde afhankelijkheid nog wel eens verstoppen in een symbool. Dat bleek ook zo te zijn: bepaalde coëfficiënten bleken indirect van de snelheid af te hangen. Daardoor werd het stelsel vergelijkingen nog ingewikkelder.”

Een groep van elf wiskundigen boog zich in februari 2011 een week lang over het probleem. Eerst analyseerden ze de vergelijkingen theoretisch. Daarna gebruikten ze deze analyse voor het bedenken van een nieuwe numerieke oplossing van het stabiliteitsprobleem, zonder de vergelijkingen in de tijd door te rekenen. Uit de analyse van de oorspronkelijke vergelijkingen bleek dat de hoek die het roer maakt met het schip een grote rol speelt. Er treedt een ingewikkeld krachtenspel op tussen het roer en de schroef, die altijd in de buurt van het roer zit. “De belangrijkste vereenvoudiging die we hebben doorgevoerd,” zegt Veerman, “is dat we die ingewikkelde krachtenbalans weglaten en modelleren door een draaibare voortstuwing aan te nemen die tegen het schip aanduwt. De positie van die draaibare voortstuwing is een vrije parameter. Het ingewikkeldere roermodel van het MARIN hebben we vervangen door ons eenvoudiger voortstuwingsmodel.”

Door alleen naar evenwichten te kijken, is de tijd als expliciete variabele uit het probleem verdwenen. Het nieuwe roermodel geeft drie eerste orde vergelijkingen in de snelheid, waarbij de afgeleide wel een tijdsafgeleide is. In het algemeen is dit stelsel vergelijkingen niet gemakkelijk oplosbaar, maar wel voor de specifieke vragen die het MARIN had gesteld: hoe zit het met de stabiliteit van rechtdoor varen en bij het draaien in een cirkel? Veerman: “Dat betekent dat we niet naar alle mogelijke evenwichten hoefden te zoeken, maar alleen naar een subklasse.”

De wiskundigen implementeerden hun theoretische analyse zowel in Matlab als in het gratis verkrijgbare software met de naam AUTO. Matlab lost de vergelijkingen direct op. AUTO is speciaal is gemaakt om te analyseren hoe de stabiliteit afhangt van een verandering van parameters (het bifurcatiegedrag). De studiegroep vond zowel in



Figuur 4.1: De krachten op een schip leveren een gekoppeld systeem van gewone differentiaalvergelijkingen met sterke niet-lineaire termen.

Matlab als in AUTO dezelfde soort bifurcaties als in de theoretische analyse. “Onze numerieke oplossing laat zien dat ons vereenvoudigde voortstuwingsmodel een goed alternatief is voor het ingewikkeldere roermodel van het MARIN”, concludeert Veerman. “We kunnen er de scheepsstabiliteit mee analyseren, en dan met name als het gaat om rechtdoor varen en in een cirkel draaien. En ten slotte kunnen we ook nog een nieuw soort oplossingen vinden: periodieke oplossingen zoals het zijwaarts of voorwaarts heen en weer bewegen van het schip. Die periodieke oplossingen vinden we als gesloten banen in de faseruimte. Dat soort oplossingen kun je in de theoretische analyse niet of in ieder geval heel moeilijk vinden.”

Als hulp bij de analyse door de wiskundigen, gaf MARIN de experimenteel bepaalde parameters van een modelschip; het model van een honderdvijftig meter lang containerschip dat in de haven van Hamburg ligt. De wiskundigen splitsten zich op in twee groepen: de ene groep ging het probleem theoretisch te lijf, de andere numeriek. Beide groepen gebruikten dezelfde experimenteel bepaalde parameters en analyseerden de stabiliteit. Beide groepen concludeerden dat het schip al bij een kleine verstoring ging afwijken van zijn rechte lijn. De wiskundigen verbaasden zich over dit resultaat, maar bij navraag bleek het precies te kloppen met het instabiele gedrag van het schaalmodel dat het MARIN had onderzocht.

Kwaliteitstoets

“De wiskundigen hebben mij op twee punten enorm verrast”, zegt MARIN-onderzoeker Ed van Dalen. “Allereerst door het feit dat er al software bleek te bestaan die evenwichtspunten berekent uit een stelsel vergelijkingen zoals het onze. Ik wist niet dat zulke software al bestond. En ten tweede doordat ze in hun vereenvoudigde analyse toch alle parameters kunnen meenemen die in de praktijk een rol spelen. Ik had van te voren gedacht dat ze ter vereenvoudiging waarschijnlijk veel parameters zouden moeten weglaten. Met het werk van de studiegroep zetten we een stap verder door ook op een wiskundige manier naar de kwaliteit van onze modellen te kijken.”

De AUTO-software is wiskundig gezien gebaseerd op ‘de impliciete functiestelling’. Die laat zien dat je onder bepaalde voorwaarden door langzaam de parameters

te veranderen nieuwe evenwichtspunten vindt. Wiskundig gezien gebeurt dat door continu lokaal een linearisering van de vergelijkingen toe te passen. Van Dalen: “Ik ben de impliciete functiestelling in mijn wiskundestudie wel tegengekomen, maar ik had er zelf nooit aan gedacht om hem op deze manier toe te passen.” Hij vergelijkt de toepassing van de impliciete functiestelling met het bepalen van de evenwichtsoplossing van een slinger: “Het slingergedrag hangt af van de massa, de lengte, de demping en eventueel de aandrijving van de slinger. Als je die parameters kent, vind je de periodieke evenwichtsoplossing in de vorm van de frequentie en de amplitude van de slinger. Op een soortgelijke manier analyseert de AUTO-software de evenwichtsoplossingen van een schip.”

Van Dalen wil het werk van de studiegroep graag voortzetten. Maar het gebruik van de analysesoftware AUTO is niet triviaal. “Je moet wel al een inzicht hebben in de differentiaalvergelijkingen, in het kiezen van goede startpunten en je moet weten welke aspecten belangrijk zijn om op te letten.” Liefst zou hij een afstudeerstudent aannemen, die zich verdiept in de AUTO-software en dan een nette procedure opschrijft hoe de scheepsstabiliteit parameter voor parameter geanalyseerd kan worden. Van Dalen: “De student zou dan met de gegevens van schaalmodellen een flink aantal schepen kunnen doorrekenen. Dat werk dient niet ter vervanging van onze modelproeven, maar als aanvulling op zowel de modelproeven als onze eigen numerieke tijdsanalyses. Het is voor ons een verbeterinstrument.”

5 Kortste routes op een analoge chip

Bennie Mols

Het ontwerp van een analoge chip gebeurt door zijn complexiteit nog steeds deels handmatig. Halfgeleiderfabrikant NXP zoekt naar een wiskundige methode om het chipontwerp verder te automatiseren.

Elektronische chips zijn wonderen van compactheid. Een digitale chip bestaat uit regelmatige blokken die allemaal ongeveer dezelfde afmetingen en vorm hebben. Dat maakt het chipontwerp relatief eenvoudig en grotendeels automatiseerbaar. Anders wordt het bij analoge chips. Op analoge chips hebben de verschillende onderdelen (blokken met complexe samenstellingen van basiscomponenten zoals weerstanden, transistoren, spoelen en condensatoren) vaak verschillende vormen. Bovendien kunnen de onderdelen elkaar storen, zeker bij gebruik van hoge signaalfrequenties.

Door de complexiteit van de analoge chips zijn de bestaande automatische ontwerpprogramma's vaak niet goed toepasbaar. Daarom gebeurt het ontwerp van analoge chips nog steeds deels handmatig. De ontwerpers gebruiken hierbij de ervaring en intuïtie die ze tijdens eerdere productontwikkeling hebben opgedaan. Ze beschouwen zichzelf soms zelfs als kunstenaars.

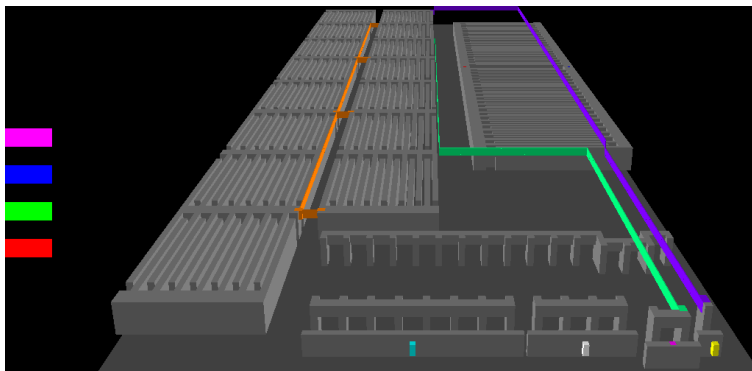
Kunnen we die intuïtie toch niet rationaliseren? vroeg chipfabrikant NXP Semiconductors N.V. uit Eindhoven (in 2006 afgesplitst van Philips) zich af. NXP is zich de laatste jaren steeds meer gaan toeleggen op het ontwerp en de fabricage van analoge chips. Ze worden vooral gebruikt in toepassingen waarbij analoge signalen uit de omgeving moeten worden opgepikt. Het analoge signaal wordt vervolgens vertaald in een digitaal signaal. Analoge chips zitten bijvoorbeeld in mobiele telefoons, in auto's en in de ov-chipkaart.

Viaducten

Wiskundige Joost Rommes is onderzoeker bij NXP en projectleider ontwerpmethoden. Hij vertelt dat hij het ontwerpprobleem wel eens heeft aangepakt, maar al snel moest concluderen dat het erg ingewikkeld werd. "Omdat wij meteen aan onze ontwerpers moeten denken, kan de probleemaanpak al snel een tunnelvisie krijgen. Ik wilde het ontwerpprobleem aan de Studiegroep Wiskunde met de Industrie voorleggen omdat de wiskundigen onbevooroordeeld naar het probleem kunnen kijken. Zij hoeven in eerste instantie geen rekening met onze ontwerpers te houden."

Een ontwerper van een analoge chip staat voor de vraag hoe hij de verschillende componenten zodanig op de chiphouder moet plaatsen dat aan een aantal randvoorwaarden wordt voldaan. Die randvoorwaarden zijn bijvoorbeeld dat de gebruikte oppervlakte minimaal is, dat de totale lengte van de verbindingen tussen componenten minimaal is, dat de verbindingen zo weinig mogelijk bochten kennen, maar ook dat componenten elkaar niet storen en dus ook voldoende afstand tot elkaar moeten houden. Bijkomende uitdaging is dat de randvoorwaarden per ontwerp (en per ontwerper) kunnen verschillen.

De analoge chip heeft een driedimensionaal karakter. Bovenop een siliciumsubstraat ligt een handvol metaallaagjes die in dikte variëren. De verbindingen tussen de componenten kunnen verschillende metaallagen gebruiken. Zo ontstaan bijvoorbeeld



Figuur 5.1: Screenshot van het ontwerp van een analoge chip. De grijze blokken stellen de elektronische componenten voor en de gekleurde lijnen de verbindingen tussen aansluitpunten op de componenten.

viaducten waar verbindingen over elkaar heen lopen. Rommes: “Onze specifieke vraag aan de studiegroep was als volgt: wij geven jullie een invoerbestand met het aantal componenten, het aantal metaallagen, de aansluitpunten op de componenten en de eis welk aansluitpunt met welk ander aansluitpunt moet worden verbonden. Kunnen jullie ons dan een algemene methode leveren die in een uitvoerbestand aangeeft waar we de componenten moeten plaatsen en hoe de verbindingen op de chip moeten lopen, rekening houdend met de gespecificeerde randvoorwaarden?”

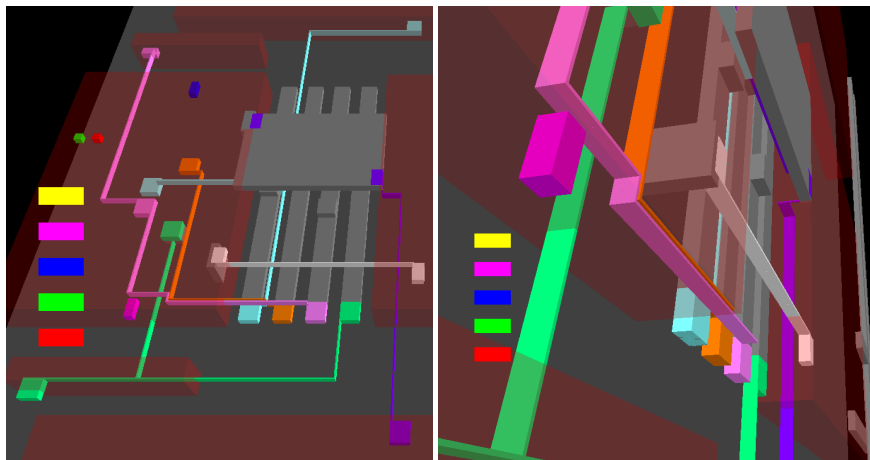
Dijkstra-algoritme

Een groep van zes wiskundigen boog zich in januari 2011 over dit probleem. “Het bleek te veel gevraagd om dit probleem binnen een week in zijn algemene vorm op te lossen”, vertelt Bas Fagginger Auer, promovendus aan de Universiteit Utrecht. “Om het probleem te vereenvoudigen, hebben we hebben besloten om de plaatsing van de componenten als gegeven te beschouwen en alleen de verbindingen als onbekenden.”

De studiegroep ontwikkelde binnen een week twee oplossingsmethoden. Allereerst een heuristische oplossingsmethode gebaseerd op het bekende Dijkstra-algoritme, dat onder meer wordt toegepast in autonavigatiesystemen. En ten tweede een methode gebaseerd op geheeltallig lineair programmeren (ILP). Voor beide methoden bekeken de wiskundigen ook nog een variant waarin meer dan twee blokken aan elkaar gekoppeld moeten worden. Hiervoor is het nodig om in plaats van paden Steiner-bomen te construeren, die de blokken onderling verbinden.

Bij de heuristische methode wordt de chip gediscrètiseerd tot een rooster. Het rooster wordt opgevat als een graaf en de roosterpunten die vallen binnen de aanwezige componenten worden verwijderd uit de graaf. Op de overgebleven graaf pasten de wiskundigen een variant van Dijkstra’s algoritme toe om kortste paden te vinden tussen twee aansluitpunten die met elkaar verbonden moeten worden.

“Het Dijkstra-algoritme zoekt zo kort mogelijke paden in een graaf”, vertelt Fagginger Auer. “Het algoritme is wiskundig eenvoudig en gemakkelijk te implementeren. Wij hebben de notie van kort’ echter vervangen door een algemene kostenfunctie, samengesteld uit de eisen die de ontwerper aan de bedrading stelt. Iedere eis heeft



Figuur 5.2: Screenshots met details uit het ontwerp van een analoge chip. Beide circuits bestaan uit vijf op elkaar gestapelde metaallagen waarin verbindingen mogen lopen. De gekleurde blokjes in het ontwerp stellen de aansluitpunten op een component voor. De gekleurde blokjes in de verticale rij aan de linkerkant geven de voorconstanten weer voor de kostenfunctie die de wiskundigen hebben gebruikt. In dit geval zijn de blokjes allemaal even lang, wat wil zeggen dat de voorconstanten allemaal even groot zijn gekozen. Zo kan een ontwerper in een oogopslag zien wat er verandert in het ontwerp wanneer een of meer voorconstanten veranderen.

zijn eigen voorconstante, waardoor de ontwerper specifieke eisen zwaarder of minder zwaar kan laten wegen door deze constanten te variëren. Bijvoorbeeld, door de constante voor de totale lengte van de bedrading groot te maken, zal het algoritme zoeken naar paden die kort zijn, maar mogelijk veel hoeken bevatten. Als daarentegen de constante voor het aantal hoeken in de bedrading groot is, staat het algoritme paden met een grotere lengte toe, als dit ervoor zorgt dat paden rechter worden.”

Verder wordt de A*-uitbreiding van het oorspronkelijke Dijkstra-algoritme gebruikt, wat ervoor zorgt dat de zoekrichting van het algoritme enigszins de goede kant op wordt gestuurd. Fagginger Auer: “Stel, dat een aansluitpunt linksboven op de chip verbonden moet worden met een aansluitpunt rechtsonder, dan is de kans klein dat de kortste route via de rechter bovenhoek verloopt. Dan sturen we de zoekrichting automatisch al naar rechtsonder. Dat scheelt veel rekentijd voor het algoritme.”

De wiskundigen implementeerden deze methode in een computerprogramma: het prototype. Dit prototype neemt een vijftal metaallagen aan waarin de verbindingen kunnen lopen, en gebruikt een rooster van duizend bij duizend punten. De oplosstijd ligt in de orde van enkele seconden tot maximaal tien seconden. “Voor ons als wiskundigen valt lastig te zeggen hoe goed de oplossing van het programma is”, zegt Fagginger Auer over het ontwikkelde prototype. “Dat komt omdat een objectieve maat ontbreekt voor wat precies een goede oplossing is.”

Lineair programmeren

Voor de tweede oplossingsmethode is het ontwerpprobleem vertaald in de context van geheeltallig lineair programmeren (ILP). Dat biedt twee voordelen ten opzichte van de

heuristische oplosmethode, aldus Fagginger Auer. “Ten eerste geeft deze methode een schatting van hoe ver een bepaalde oplossing af zit van het optimum. Ten tweede loopt deze methode veel meer mogelijke paden af. De heuristische methode legt de kortste paden als eerste aan en daarna steeds langere paden. Je kunt je echter voorstellen dat het soms beter is om eerst een langer pad aan te leggen en daarna een korter pad. Dat kan met de ILP-methode omdat deze methode paden als het ware aan en uit kan zetten. De methode van de column generation berekent dan welke paden de grootste toename van de kwaliteitsfunctie geven. Alleen die paden worden verder geanalyseerd.”

De ILP-methode kost meer rekenwerk dan de heuristische methode, maar is ook preciezer. Of die precisie de moeite loont vergeleken met een toename van het rekenwerk, kan Fagginger Auer niet zeggen. “We hebben op papier een analyse gegeven van hoe ILP het chipontwerp kan uitrekenen, maar de studieweek was te kort om deze methode ook te implementeren in een computerprogramma, zoals we bij de heuristische methode wel hebben gedaan.”

Prototype

“Mijn verwachtingen zijn ruimschoots overtroffen”, zegt NXP-onderzoeker Joost Rommes over de resultaten van de studiegroep. “In de eerste plaats hebben de wiskundigen een mooi prototype gemaakt waarover ook onze ontwerpers enthousiast zijn. De ontwerpers kunnen het prototype niet meteen gebruiken dat was ook niet het doel maar het is wel een mooi startpunt om de automatisering van het chipontwerp verder te ontwikkelen. De methoden die de wiskundigen hebben ontwikkeld zijn algemeen. Bij een gegeven invoerbestand, berekenen ze een uitvoerbestand. In de tweede plaats kunnen we de analyse en het prototype gebruiken in onderhandelingen met verkopers van ontwerpsoftware. Met het prototype in de hand kunnen wij bijvoorbeeld aantonen wat minstens haalbaar is en daarmee onze eisen aan de softwarefabrikant hard maken.”

NXP maakt van een product vaak meerdere varianten, bijvoorbeeld omdat frequentiebanden in de praktijk nogal eens worden veranderd, waardoor een afnemer vraagt om een nieuwe chipvariant. Bij het ontwerp van die varianten wil het bedrijf liefst een bepaalde ontwerpbasis hergebruiken. Dat levert namelijk minder werk. “De resultaten van de studiegroep zijn voor ons ook een aanzet om een zogeheten geparameteriseerd ontwerp te ontwikkelen”, zegt Rommes. “Met geparameteriseerd bedoelen we dat we bij het veranderen van het chipontwerp een aantal parameters aanpassen, bijvoorbeeld het aantal transistoren dat in een blok zit. Idealiter willen we een computerprogramma waarin we voor een nieuwe variant van eenzelfde product alleen een aantal parameters hoeven te veranderen, waarna het programma met een druk op de knop het nieuwe chipontwerp geeft. Zo kunnen we sneller inspelen op de vragen van onze klanten.”

6 Was de doodrijder ook een hardrijder?

Bennie Mols

Het Nederlands Forensisch Instituut wordt regelmatig gevraagd om de snelheid te schatten die een auto had voordat hij botste met een andere auto. Wat is daarvoor de beste methode, gegeven alle onzekerheden in de relevante botsingsparameters?

Midden op de snelweg staan twee ingedeukte auto's. Duidelijk total loss door een kop-staart-botsing. Alle inzittenden zijn net zwaar gewond met ambulances afgevoerd. Er staat een remspoor van dertig meter op de weg. Het lijkt er op dat ergens halverwege de botsing heeft plaatsgevonden. Als de achteropkomende bestuurder te hard heeft gereden, dan kan de rechtbank hem veroordelen tot een gevangenisstraf. Maar dan moet ze wel voldoende bewijs hebben. Het Nederlands Forensisch Instituut (NFI) wordt daarom gevraagd om zo nauwkeurig mogelijk te bepalen hoe hard de auto reed voordat hij begon te remmen of voordat hij botste, in het geval dat hij niet remde.

In theorie is dit een eenvoudig terugrekenprobleem uit de klassieke mechanica dat met de wetten van behoud van impuls en behoud van energie opgelost kan worden: bepaal uit een bekende eindtoestand de onbekende begintoestand. Voor dit terugrekenen is echter veel informatie nodig: onder andere de lengte van het remspoor, de massa van het voertuig voor de botsing, de vertraging tijdens de remweg en de hoeveelheid energie die in het vervormen van de auto's tijdens de botsing is gaan zitten. Typisch is er te weinig informatie over de eindtoestand om deze nauwkeurig te kunnen reconstrueren.

“De lengte van de remweg is nauwkeurig te meten”, vertelt NFI-onderzoeker Aart Spek. “Maar hoe zit het met de remvertraging? Op stroef asfalt kan dat wel 9 m/s^2 zijn, terwijl het op een glad wegdek 6 m/s^2 kan zijn. Maar hoe stroef was het werkelijke asfalt? En wat was de massa van het voertuig voor de botsing? De inzittenden liggen in het ziekenhuis, dus hun massa moet worden geschat. Hetzelfde geldt voor de hoeveelheid brandstof die na de botsing uit de tank is weggelekt. De hoeveelheid energie die in de botsing is gaan zitten, schatten we zo goed mogelijk door de vervorming te vergelijken met die van gecontroleerde botsingsproeven waarvan de gegevens exact bekend zijn. Maar de onzekerheid in de vervorming is een stuk groter dan die in de lengte van het remspoor.”

Het NFI moet de extra informatie die nodig is om het terugrekenprobleem op te lossen zo goed mogelijk zien te schatten. Dat leidt tot een statistische verdeling voor de waarde die elke botsingsparameter kan aannemen. De state-of-the-art-methode waarmee het NFI dit probleem oplost heet MC Crash. MC Crash combineert een deterministisch fysisch botsingsmodel (PC Crash), dat de wetten van behoud van impuls en energie gebruikt, met een statistische verdeling voor de botsingsparameters. Met de Monte Carlo-methode worden enkele miljoenen parametersets getrokken uit de statistische verdeling van alle parameters. Voor elke getrokken parameterset voert het NFI een simulatie uit in het fysische botsingsmodel.

PC Crash rekent nu voor elke getrokken parameterset een eindtoestand uit. Wanneer de berekende eindtoestand niet past bij de ongevalssporen, dan wordt de gebruikte parameterset verworpen en anders wordt hij geaccepteerd als een mogelijke parameterset van de echte botsing. De precieze selectiecriteria voor acceptatie of verwerping van een parameterset worden door experts gekozen en kunnen van geval tot geval een



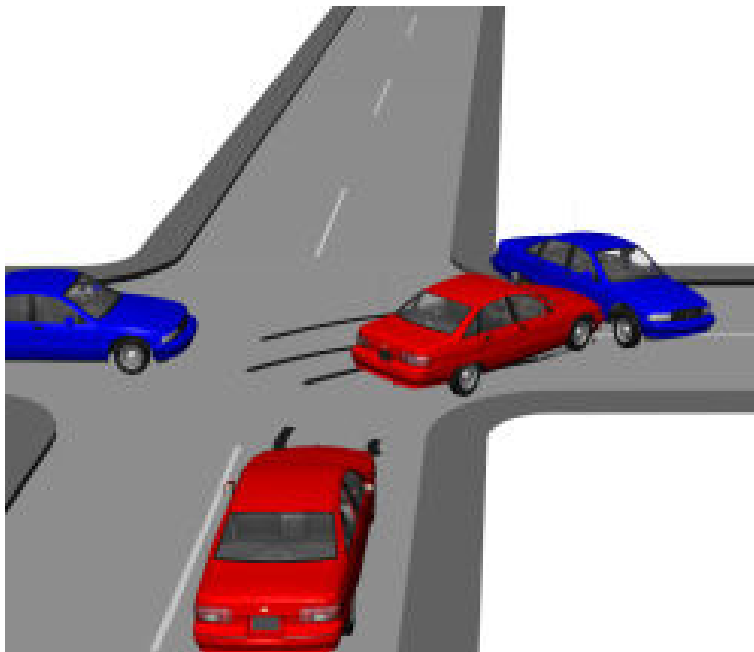
beetje verschillen. Ze bevatten bovendien meestal maar een deel van de beschikbare informatie. Zo kan een expert er voor kiezen om alleen de eindpositie van slechts een auto mee te nemen, wanneer de eindpositie van de andere auto onzeker is, bijvoorbeeld omdat hij na de botsing tot stilstand is gekomen in een greppel in plaats van op een rechte weg door de wrijving met het asfalt tot stilstand te zijn gekomen.

“Het probleem dat we aan de studiegroep hebben voorgelegd”, vertelt Spek, “gaat over de keuze van die selectiecriteria. Wat is de invloed van de selectiecriteria op de betrouwbaarheid waarmee we de beginsnelheid van de auto schatten? Leidt een bepaalde manier van schatten bijvoorbeeld tot een te conservatieve inschatting van de betrouwbaarheid van de beginsnelheid? Wij hebben het idee dat onze eigen aanpak tot een conservatieve schatting van het betrouwbaarheidsinterval voor de beginsnelheid leidt. Klopt dat idee?”

Statistische toets

“Een week was veel te kort om de NFI-methode volledig te doorgronden”, vertelt Wouter Kager, universitair docent kansrekening aan de Vrije Universiteit Amsterdam. “Er zaten voor ons te veel technische kanten aan, die te maken hebben met hoe het NFI de kansverdelingen van allerlei parameters van de auto’s bepaalt. Hun methode is in hoge mate een ingenieursmethode, eerder dan een zuiver wiskundige methode. En dat maakte het voor ons te lastig om op hun oorspronkelijke vraagstelling in te gaan. Wij hebben daarom het probleem gemodelleerd op een manier die zoveel mogelijk abstraheert van alle technische details, en zijn zo op een alternatieve methode uitgekomen.”

Het NFI wil weten wat de kansverdeling voor de beginsnelheid is, gegeven de meting van de toestand na de botsing. Het wiskundige model zegt echter iets over de verdeling van de output, gegeven een input. Volgens de Bayesiaanse statistiek is er voor de omdraaiing van deze argumentatie een prior nodig: een eerste aanname

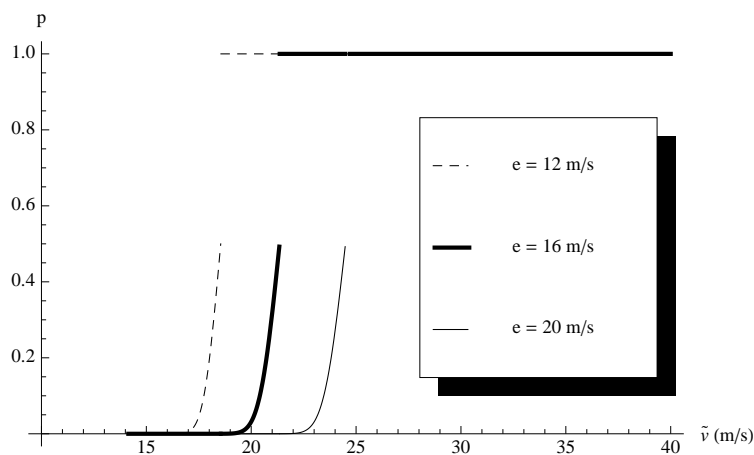


over de snelheidsverdeling vóór de botsing. Het NFI gebruikt inderdaad zo'n prior, maar de wiskundigen vonden het problematisch om er zomaar eentje aan te nemen, zonder daar gegronde redenen voor te hebben. Daarom bekeken ze een alternatieve methode die geen prior nodig heeft: een statistische toets, de Statistical Significance Testing.

Kager: "Onze toetsingsmethode berekent voor elke inputsnelheid hoe onwaarschijnlijk het is dat we een eindtoestand meten zoals die is gemeten. Die waarschijnlijkheid drukken we uit in een p-waarde. De p-waarde van een bepaalde hypothese over de inputsnelheid is de maximale kans op een meting die minstens zo extreem is als wat er werkelijk is gemeten onder die hypothese. Met deze methode tasten we als het ware het hele snelheidsveld van mogelijke beginsnelheden af."

Kager en zijn collega-wiskundigen hebben voor een eenvoudig rekenvoorbeeld de p-waarde berekend van de hypothese dat de beginsnelheid onder een bepaalde maximale waarde ligt, als functie van die maximale waarde. Dat levert typisch een grafiek op waarbij p heel lang 0 blijft en opeens vrij snel toeneemt naar 1. Vervolgens blijft p ook 1, hoe hoog de snelheid ook wordt. Kager: "Stel, de echte botsingssnelheid was 50 km/uur. Dan is p onder de aanname van een maximale beginsnelheid van 10 km/uur heel klein. Bijvoorbeeld vanaf 40 km/uur zie je p dan snel stijgen. Bij bijvoorbeeld 60 km/uur is p gelijk aan 1 geworden. De grafiek suggereert dat de echte snelheid bij die scherpe overgang ligt, maar dat hoeft niet zo te zijn. Het zegt alleen dat wat er is gemeten erg onwaarschijnlijk is onder de aanname van een lagere botsingssnelheid."

Ten slotte analyseerde de studiegroep ook nog hoe groot de invloed is van het feit dat sommige botsingsparameters (zoals de vervormingsenergie tijdens de botsing) veel onzekerder zijn dan andere (zoals de lengte van de remweg). "We vonden hier dat wat je ook precies verwacht: wanneer bijvoorbeeld de vervormingsenergie klein is ten opzichte van de energie vóór de botsing, dan is het niet erg dat de onnauwkeurigheid



Figuur 6.1: P-waarden als functie van de snelheid

in de vervormingsenergie vrij groot is. Maar wanneer de vervormingsenergie groot is ten opzichte van de energie voor de botsing, is het wel erg.”

Wel of geen prior—dat is de vraag

“De wiskundigen geven aan dat het niet mogelijk is om op zuiver wiskundige wijze in de context van het botsingsprobleem de waarschijnlijkheid van de beginsnelheid te bepalen”, vat NFI-onderzoeker Aart Spek de belangrijkste conclusie in zijn eigen woorden samen. “Onze oorspronkelijke vraag aan hen valt daarom binnen een methode die zij niet geëigend vinden. Dat is de reden dat ze niet hebben kunnen ingaan op de oorspronkelijke vraag die wij aan ze hadden. Dat vind ik jammer, maar vanuit die optiek wel begrijpelijk.”

Het NFI begint met de aanname dat elke beginsnelheid a priori even waarschijnlijk is. Dat is een keuze. Andere keuzen zijn ook mogelijk. Uit onderzoek is bijvoorbeeld bekend dat elke snelheidstoename met vijf kilometer per uur de ongevalskans ongeveer verdubbelt. Op grond van het gegeven dat de auto heeft gebotst, zou je dus ook kunnen redeneren dat hogere snelheden op voorhand (a priori) waarschijnlijker te achten zijn dan lagere snelheden. Dat zou dan dus een andere prior opleveren dan wij gebruiken. In onze rapportage aan de rechtbank geven we daarom duidelijk aan welke prior wij gebruiken, en ook dat we op verzoek van een andere prior uit zouden kunnen gaan.

“Als ik het verhaal van de wiskundigen goed begrijp,” zegt Spek, “dan zeggen ze het volgende: wij zoeken een methode die onafhankelijk is van een aanname over de beginsnelheid, de prior. Een prior is dat wat men op voorhand meent te weten over de vraag die wordt gesteld: de snelheid. De praktijk in de forensische wereld is dat wij informatie geven aan de rechtbank en dat de rechtbank vervolgens haar mening over de verdachte update. Die mening kan gebaseerd zijn op kennis die de rechtbank over de verdachte heeft. Is het iemand die nooit eerder is veroordeeld voor te hard rijden of is iemand al vaker veroordeeld voor te hard rijden? Vanuit wiskundig oogpunt

kan de kritiek luiden: hoe kan de rechtbank nou een toepasselijke prior hebben? Dat is een relevante vraag, waarover ook wel wordt gediscussieerd, maar in het huidige juridische model werkt het wel zo. Wij als NFI acteren binnen dat model.”

Spek: “De methode die de studiegroep ontwikkeld heeft, is een minimalisatiemethode. We hebben indertijd niet voor dergelijke methoden gekozen vanwege het gevaar te blijven hangen bij een lokaal minimum en het lokale minimum aan te zien voor een globaal minimum. Dat was een decennium terug—goede algoritmen voor globale optimalisatie liepen toen spaak op enorme rekentijden. We hebben gesproken met Wouter Kager en Thomas Rot van de studiegroep om een nog beter inzicht te krijgen wat hun bevindingen betekenen voor onze methode. Nu onderzoeken we of en in welke mate we de bevindingen van de studiegroep kunnen gebruiken om onze methode verder te verbeteren. We zijn het er in elk geval volledig over eens dat we in onze rapporten duidelijk en transparant moeten zijn over de gebruikte prior. ”

Optimal Flood Control

*Peter Dickinson** *Joost Hulshof†* *André Ran†*
Majid Salmani‡ *Martijn Zaal†*

Abstract

A mathematical model for optimal control of the water levels in a chain of reservoirs is studied. Some remarks regarding sensitivity with respect to the time horizon, terminal cost and forecast of inflow are made.

1 Introduction

For the Study Week Mathematics and Industry a problem was proposed by Deltares concerning the computation of optimal controls for the settings of hydraulics. The long term goal that Deltares has is to provide methods for optimal control of the settings of the hydraulics in the Dutch river system as a whole. At present each of the hydraulics is treated separately and not in combination. A “toy problem” was provided by Deltares to use in the study week. See [32]. This simple problem consists of a chain of four reservoirs, with an inflow into the first reservoir which is predictable on a reasonable time-scale. The model will be explained in the first section of this paper.

At present there is a method to compute the optimal control in use at Deltares. The questions posed by Deltares were two-fold: is this a good method, and are there ways to improve on it? In addition, it would be very nice if there was an efficient way to adjust optimal control to changing forecasts of inflow. During the study week the team also discussed issues connected to the modeling of the system.

2 The model

In this section, the problem will be modeled and formulated as an optimal control problem.

The water levels in a chain of reservoirs can be represented by a vector $\mathbf{x} = (x^1, \dots, x^m)$. A differential equation for \mathbf{x} can be derived by studying the flow of water through the chain of reservoirs. The water flowing out of a certain reservoir can be partially controlled by a hydraulic structure, corresponding to a control $\mathbf{u} = (u^1, \dots, u^m)$. If the water level rises above a certain critical level x_{cr}^j , extra

*Rijksuniversiteit Groningen

†VU University Amsterdam

‡University Bremen

water will flow out of the reservoir through a so-called spillway. Hence, the volume of water flowing out of a reservoir is given by

$$u^j - g^j(x - h_{cr})$$

where g^j is a function satisfying $g(y) = 0$ for $y < 0$. The water flowing into a reservoir is simply the flow out of the previous reservoir. The flow of water into the first reservoir is denoted by p , which is, for the time being, assumed to be given. Taking the surface area of the reservoirs, which is assumed to be independent of the water level, into account, this results in the following differential equation for \mathbf{x} .

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, p) \\ \mathbf{x}(0) = \mathbf{x}_0, \end{cases} \quad (1)$$

where \mathbf{x}_0 is a given initial condition, and \mathbf{f} is given by

$$f^j(\mathbf{x}, \mathbf{u}, p) = \begin{cases} \frac{1}{A^1} (p - u^1 - g^1(x^1 - h_{cr}^1)), & \text{if } j = 1, \\ \frac{1}{A^j} (u^{j-1} + g^{j-1}(x^{j-1} - h_{cr}^{j-1}) - u^j - g^j(x^j - h_{cr}^j)), & \text{otherwise.} \end{cases} \quad (2)$$

Here, the numbers A^j indicate the surface area of the reservoirs, which are assumed to be independent of the water levels for simplicity.

There are certain restrictions to the controlled outflow. Each hydraulic structure has a maximum flow of u_{\max}^j , and the setting of the structures can only be changed at certain predetermined moments. More precisely, $\mathbf{u}(t) = \mathbf{u}_i$ for $t \in (\frac{i-1}{n}, \frac{i}{n}]$.

The objective is to keep the water levels as close as possible to a given set point level $h_{sp}^j < h_{cr}^j$, minimizing the spillover, while adjusting the hydraulic structures a little as possible. More precisely, the objective function

$$J(\mathbf{u}, \mathbf{x}) = \phi(\mathbf{x}(T)) + \int_0^T L(\mathbf{x}(t), \mathbf{u}(t)) dt + \frac{1}{2} \sum_{i=1}^n \|\mathbf{u}_i - \mathbf{u}_{i-1}\|^2 \quad (3)$$

is minimized, where \mathbf{u}_0 is the (given) initial setting of the hydraulic structures, and L represents the penalty for deviation from the set point level and use of the spillway. The function ϕ indicates terminal costs and is not specified here.

In this particular model, specific choices for L and g will be used:

$$L(\mathbf{x}, \mathbf{u}) := \sum_{j=1}^m w_{sp}^j (x - x_{sp})^2 + w_{cr}^j ((x - x_{cr})^+)^2, \quad (4)$$

$$g^j(y) := C^j ((x - x_{cr})^+)^{\frac{3}{2}}. \quad (5)$$

Putting all equations together, the optimal control problem is

$$\begin{cases} \text{minimize} & J(\mathbf{u}, \mathbf{x}) \\ \text{under} & \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, p) \\ & \mathbf{x}(0) = \mathbf{x}_0 \\ & 0 \leq u_i^j \leq u_{\max}^j \quad (j = 1, \dots, m) \end{cases} \quad (6)$$

The problem as outlined above may be found in [32], where also a case study was undertaken for the case of a chain of four reservoirs.

3 Lagrange multipliers

3.1 Optimality conditions

Ignoring the range constraint on $\mathbf{u}(t)$ for the time being, optimality conditions for (6) can be derived by studying the critical points of the extended Lagrange functional

$$\mathcal{L}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, p) = \phi(\mathbf{x}(T)) + \int_0^T L(\mathbf{x}, \mathbf{u}) + \boldsymbol{\lambda}(\mathbf{f} - \dot{\mathbf{x}}) dt + \frac{1}{2} \sum_{i=1}^n \|\mathbf{u}_i - \mathbf{u}_{i-1}\|^2, \quad (7)$$

where $\boldsymbol{\lambda}$ is interpreted as a row vector. For notational convenience, introduce $H := L + \boldsymbol{\lambda}\mathbf{f}$.

Choosing a smooth variation $\boldsymbol{\xi}$ for \mathbf{x} ,

$$\begin{aligned} \left\langle \frac{\partial \mathcal{L}}{\partial \mathbf{x}}, \boldsymbol{\xi} \right\rangle &= \frac{d\phi}{d\mathbf{x}}(\mathbf{x}(T))\boldsymbol{\xi}(T) + \int_0^T \frac{\partial H}{\partial \mathbf{x}} \boldsymbol{\xi} - \boldsymbol{\lambda} \dot{\boldsymbol{\xi}} dt \\ &= \left(\frac{d\phi}{d\mathbf{x}}(\mathbf{x}(T)) - \boldsymbol{\lambda} \right) \boldsymbol{\xi}(T) + \int_0^T \left(\frac{\partial H}{\partial \mathbf{x}} + \dot{\boldsymbol{\lambda}} \right) \boldsymbol{\xi} dt + \boldsymbol{\lambda}(0)\boldsymbol{\xi}(0), \end{aligned} \quad (8)$$

where derivatives with respect to vectors are interpreted as row vectors. Note that the last term vanishes because there is an initial condition for \mathbf{x} . Hence, requiring that the variation of \mathcal{L} with respect to \mathbf{x} is zero results in a terminal value problem for $\boldsymbol{\lambda}$:

$$\begin{cases} \dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \mathbf{x}} \\ \boldsymbol{\lambda}(T) = \frac{d\phi}{d\mathbf{x}}(\mathbf{x}(T)) \end{cases} \quad (9)$$

Since \mathbf{u} is required to be piecewise constant, the variation with respect to \mathbf{u} is actually a partial derivative:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{u}_i} = \int_{\frac{i-1}{n}}^{\frac{i}{n}} \frac{\partial H}{\partial \mathbf{u}_i} dt - \mathbf{u}_{i+1} + 2\mathbf{u}_i - \mathbf{u}_{i-1}, \quad (10)$$

if $i < n$. In case $i = n$, the last three terms become $\mathbf{u}_i - \mathbf{u}_{i-1}$. Equivalently, one could introduce an artificial variable \mathbf{u}_{n+1} and require that $\mathbf{u}_{n+1} = \mathbf{u}_n$.

As expected, since $\frac{\partial H}{\partial \boldsymbol{\lambda}} = \mathbf{f}$, setting the variation of \mathcal{L} with respect to $\boldsymbol{\lambda}$ equal to zero gives back (1).

All variations together result in the optimality conditions

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f} \\ \dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \mathbf{x}} \\ \mathbf{u}_{i+1} - 2\mathbf{u}_i + \mathbf{u}_{i-1} = \int_{\frac{i-1}{n}}^{\frac{i}{n}} \frac{\partial H}{\partial \mathbf{u}_i} dt \\ \mathbf{x}(0) = \mathbf{x}_0 \\ \boldsymbol{\lambda}(T) = \frac{d\phi}{d\mathbf{x}}(\mathbf{x}(T)) \\ \mathbf{u}_{n+1} - \mathbf{u}_n = \mathbf{0} \end{cases} \quad (11)$$

Note that this is almost a coupled system of ordinary differential equations: only the equation for u is discretized. The artificial condition $\mathbf{u}_{n+1} - \mathbf{u}_n = \mathbf{0}$ can be considered to be the discretized analog of the terminal condition $\dot{\mathbf{u}}(T) = \mathbf{0}$.

3.2 Deltares's approach

A very intuitive strategy for solving (6) is to first solve the initial value problem for \mathbf{x} in terms of the control \mathbf{u} , substituting the solution into the criterion J , and minimizing the resulting function of \mathbf{u} . More precisely, the dependence of \mathbf{x} on \mathbf{u} can be written and used to study the optimum, at least formally:

$$\Phi(\mathbf{u}) := \phi(\mathbf{x}(T; \mathbf{u})) + \int_0^T L(\mathbf{x}(t; \mathbf{u}), \mathbf{u}(t)) dt + \frac{1}{2} \sum_{i=1}^n \|\mathbf{u}_i - \mathbf{u}_{i-1}\|^2 \quad (12)$$

where $\mathbf{x}(\cdot; \mathbf{u})$ is the solution of the initial value problem (1).

The partial derivatives of Φ with respect to the \mathbf{u}_i 's now involve an extra term representing the \mathbf{u} -dependence.

$$\begin{aligned} \frac{\partial \Phi}{\partial \mathbf{u}_i} &= \frac{d\phi}{d\mathbf{x}}(\mathbf{x}(T; \mathbf{u})) \frac{\partial \mathbf{x}(T; \mathbf{u})}{\partial \mathbf{u}_i} + \int_0^T \frac{\partial L}{\partial \mathbf{x}} \frac{\partial \mathbf{x}(t; \mathbf{u})}{\partial \mathbf{u}_i} dt \\ &\quad + \int_{\frac{i-1}{n}}^{\frac{i}{n}} \frac{\partial L}{\partial \mathbf{u}_i} dt - \mathbf{u}_{i+1} + 2\mathbf{u}_i - \mathbf{u}_{i-1}, \end{aligned} \quad (13)$$

Assuming some smoothness, $\frac{\partial \mathbf{x}(t; \mathbf{u})}{\partial \mathbf{u}_i}$ satisfies the following initial value problem

$$\begin{cases} \frac{\partial}{\partial t} \frac{\partial \mathbf{x}(t; \mathbf{u})}{\partial \mathbf{u}_i} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}(t; \mathbf{u})}{\partial \mathbf{u}_i} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \chi_{(\frac{i-1}{n}, \frac{i}{n}]}(t), \\ \frac{\partial \mathbf{x}(0; \mathbf{u})}{\partial \mathbf{u}_i} = \mathbf{0}. \end{cases} \quad (14)$$

Introducing λ as the solution of the terminal value problem (9), the first integral in (13) can be simplified:

$$\begin{aligned} \int_0^T \frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}(t; \mathbf{u}), \mathbf{u}(t)) \frac{\partial \mathbf{x}(t; \mathbf{u})}{\partial \mathbf{u}_i} dt &= \int_0^T \left(\frac{\partial H}{\partial \mathbf{x}} - \lambda \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right) \frac{\partial \mathbf{x}(t; \mathbf{u})}{\partial \mathbf{u}_i} dt \\ &= - \int_0^T \left(\dot{\lambda} + \lambda \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right) \frac{\partial \mathbf{x}(t; \mathbf{u})}{\partial \mathbf{u}_i} dt \\ &= - \int_0^T \left(\dot{\lambda} + \lambda \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right) \frac{\partial \mathbf{x}(t; \mathbf{u})}{\partial \mathbf{u}_i} dt \end{aligned} \quad (15)$$

Integrating by parts, the term involving $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ drops out, leaving only a term involving $\frac{\partial \mathbf{f}}{\partial \mathbf{u}}$ and initial and terminal values:

$$\int_0^T \frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}(t; \mathbf{u}), \mathbf{u}(t)) \frac{\partial \mathbf{x}(t; \mathbf{u})}{\partial \mathbf{u}_i} dt = - \frac{d\phi}{d\mathbf{x}}(\mathbf{x}(T; \mathbf{u})) \frac{\partial \mathbf{x}(T; \mathbf{u})}{\partial \mathbf{u}_i} + \int_{\frac{i-1}{n}}^{\frac{i}{n}} \lambda \frac{\partial \mathbf{f}}{\partial \mathbf{u}} dt \quad (16)$$

Substituting this back into (13), it follows that

$$\frac{\partial \Phi}{\partial \mathbf{u}_i} = \int_{\frac{i-1}{n}}^{\frac{i}{n}} \frac{\partial H}{\partial \mathbf{u}_i} dt - \mathbf{u}_{i+1} + 2\mathbf{u}_i - \mathbf{u}_{i-1} = \frac{\partial \mathcal{L}}{\partial \mathbf{u}_i}(\mathbf{x}(\cdot; \mathbf{u}), \mathbf{u}, \boldsymbol{\lambda}(\cdot; \mathbf{u}), p) \quad (17)$$

where \mathbf{x} and $\boldsymbol{\lambda}$ solve (1) and (9), respectively. Apparently, the derivative of Φ can be computed using the Lagrange multipliers.

This can be exploited when numerically solving the optimal control problem: given an initial guess for \mathbf{u} , one can find the derivative of Φ , which can, of course, be used to improve the guess, by computing the Lagrange multipliers. The efficiency of this method obviously also depends on the quality of the initial guess for \mathbf{u} . Another appealing property of this approach is that it is easy to add the bounds on \mathbf{u} . Indeed, at each step of the algorithm one checks to see whether the newly computed \mathbf{u} satisfies the bounds. If not, \mathbf{u} is at either its maximum or its minimum, and one follows the boundary of the feasible set until the gradient is pointing into the feasible set again.

The method explained above is to some extent classical in optimal control theory, see, e.g., [3, 9, 16].

3.3 Summary of the algorithm

Starting from an initial guess of \mathbf{u} one solves the system of equations forward in time for \mathbf{x} and determines $\mathbf{x}(T)$ this way. Then, one solves for $\boldsymbol{\lambda}(t)$ backwards in time. Using these, one finds $\frac{\partial \Phi}{\partial \mathbf{u}_i}$, and adjust \mathbf{u} using this. Then one repeats the cycle, until convergence is reached.

Obviously, the selection of an initial guess of \mathbf{u} is important in case the algorithm converges slowly, or takes a lot of time to execute. One may use the fact that for the case of no disturbance the system is linear and the cost function quadratic, in which case direct application of standard optimal LQ control (see e.g. [9]) gives us an initial guess. However, as it turns out, this initial guess may not be feasible in the sense that it violates $0 \leq u_i^j \leq u_{max}^j$. In the application to the toy problem this was observed in practice. Just taking the optimal LQ-control input and multiplying it with the characteristic function of $[0, u_{max}]$ gives an initial guess that may be better than just taking the zero input as initial guess.

4 Some remarks

Implementing the method above for the extremely simple case of one reservoir was carried out as a bachelor thesis project by one of the students (Hidde Kok) at VU University. The case considered was that of one reservoir with area $A = 60000$, with critical level $h_{cr} = 0.2$, and set-point water level at $h_{sp} = 0$. The weights in the cost function were set to $w_{sp} = 1$ and $w_{cr} = 10$. The final state was penalized with $2(x(T) - h_{sp}) + 2 \max(0, x(T) - h_{cr})$. The perturbation p was set at $p = 50$ for the duration of the period of 20 to 50 minutes from the start. The program did not pose any restriction on the values of the input. We compared with the maximum level of u set to $u_{max} = 20$, and with the minimum level $u_{min} = 0$.

The student's program was run for two cases: one with time horizon two hours, one with time horizon ten hours. The two graphs below show the results.

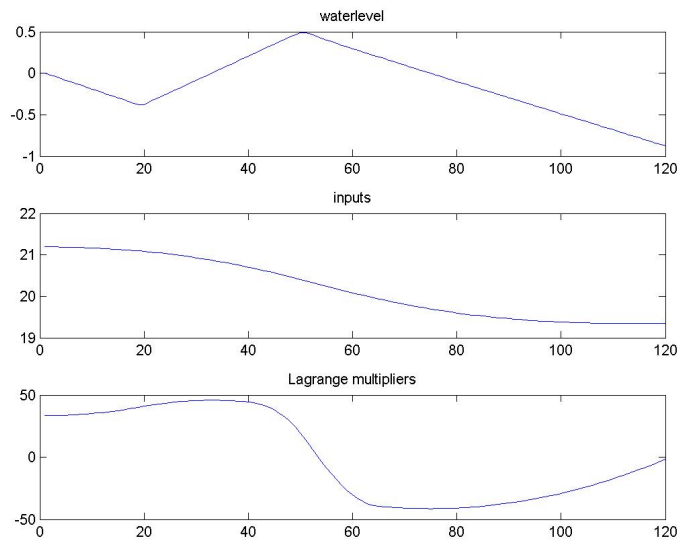


Figure 1: Two hour time horizon, no bounds on the input.

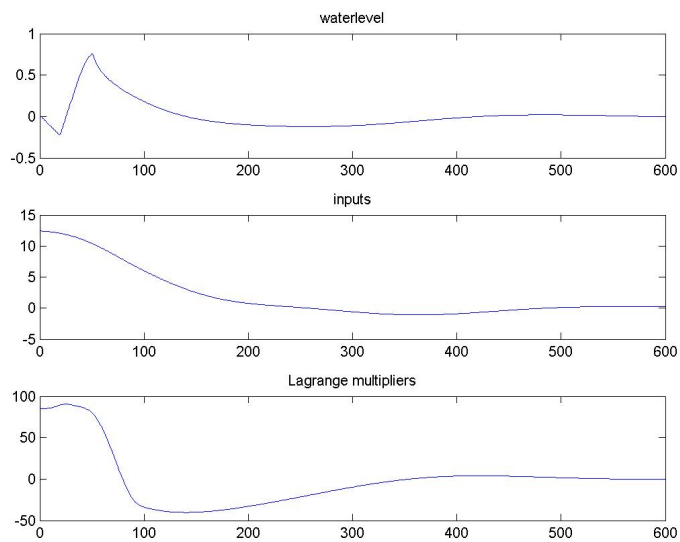


Figure 2: Ten hour time horizon, no bounds on the input.

It is obvious from figure 1 and 2 that the time horizon has a serious effect on the optimal input. This is caused by the fact that the influence of the cost on the final state is spread over a larger period. Also note that the two hour time horizon is not enough to restore the water level to the set point, and that for the two hour time horizon the inputs are above the maximum level, whereas for the ten hour time horizon this never occurs. However, for the ten hour time horizon, the input is (slightly) below zero for a considerable amount of the time. In addition, in the ten hour case we see that the peak of the water level is higher. This is caused by the fact that the cost puts more weight (due to the larger time period) on the deviation from the set point level.

In figures 3 and 4 the same settings are considered, however, now with bounds on the input taken into account. Observe that again we see that there is a substantial difference between time horizon two hours and time horizon ten hours. Comparing to the case without bounds on the input we see that even with a large time horizon the set point level is not reached. This is due to the fact that both the minimal value of the control and the reference level of the forecasted inflow are zero. In practice, however, this will not be the case: the forecasted inflow will always be positive, whereas the minimal value of the control remains zero. This difference might have been taken into account in the model by taking a negative minimal value for the control, or, alternatively, taking a positive reference level for the forecasted inflow. It can be seen from figure 2 that a small negative control is needed to return to the set point level if the water level drops below h_{sp} after the disturbance in the inflow has passed. We did not take this into account in our example in order to stay close to the model in [32].

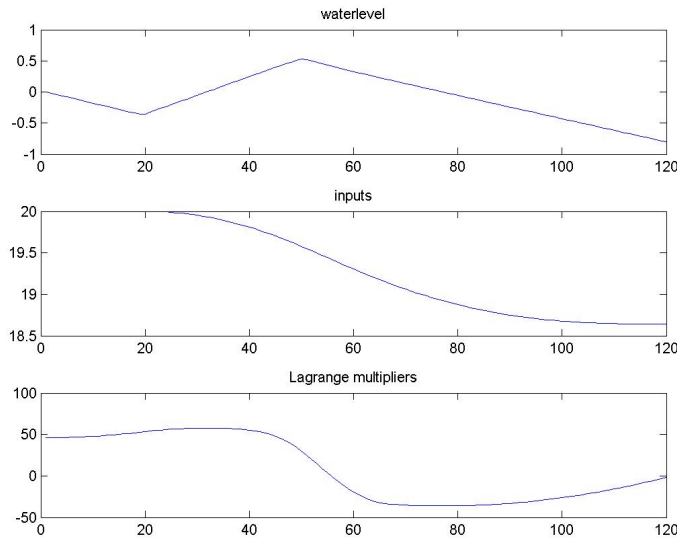


Figure 3: Two hour time horizon taking into account the bounds on the input.

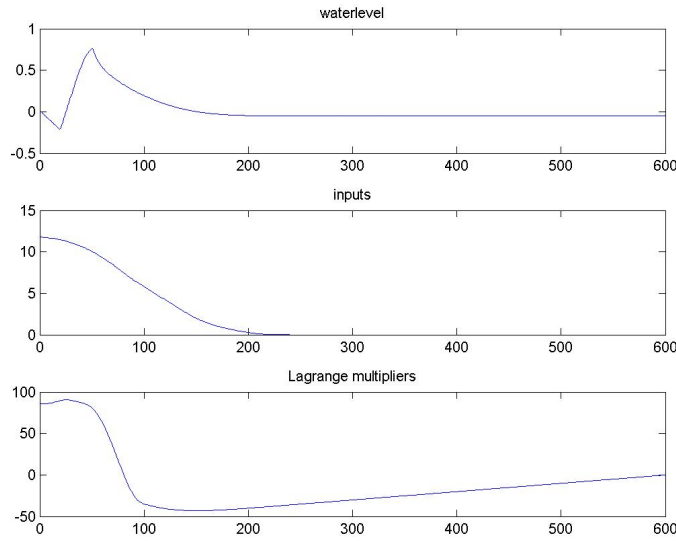


Figure 4: Ten hour time horizon taking into account the bounds on the input.

5 NLP formulation

In order to solve (6), there are some numerical methods using nonlinear programming techniques, see [10, 11, 9, 7]. These methods use a suitable discretization of the control problem by which it is transcribed into a parametric NLP problem [4, 8, 18]. The discretization methods can be divided in two general categories: Full-discretization and Partial-discretization. In the first method, all state and control variables are discretized, but in the second method, only control variables are discretized. Indeed, the states will be determined by integration. The nature of the problem suggests that partial discretization should be used, as demonstrated in section 2. The state approximations $x_i \in \mathbb{R}^n$ of the values $x(t_i)$ can be achieved recursively as functions of the control variables by an integration scheme such as forward Euler or Runge-Kutta approximation.

The problem (6) defines an NLP of the general form

$$\begin{cases} \text{minimize} & F(z, p) \\ \text{under} & G_i(z, p) = 0 \quad (i = 1, \dots, N_e) \\ & G_j(z, p) \leq 0 \quad (j = 1, \dots, N_i) \end{cases} \quad (18)$$

where $F : \mathbb{R}^{N_z} \times P \rightarrow \mathbb{R}$, $G_i, G_j : \mathbb{R}^{N_z} \times P \rightarrow \mathbb{R}$, N_e and N_i are the numbers of equality and inequality constraints, respectively. Note that the number of decision variables in this NLP(p) equals $N_z = mn$. We shall study the differential properties of optimal solutions to the perturbed problem NLP(p) and the related optimal values of the objective function with respect to p in a neighborhood of nominal parameter p_0 .

5.1 Parameter Sensitivity Analysis of NLP(p)

Since perturbations are unavoidable, only having knowledge about the nominal optimal solution is not enough and analysis of the effects of perturbations is necessary. So far we have transformed the optimal control problem with parameter p into the NLP(p) problem (18). After solving (6), independent of the discretization technique, we know the set and the number of active constraints N_a . Assume that $G^a(z^*, p_0)$ denotes the collection of the active constraints at point (z^*, p_0) . Similar to the sections above, the Lagrangian function is defined by

$$\mathcal{L}(z, \lambda, p) = F(z, p) + \lambda^T G^a(z, p). \quad (19)$$

The sufficient conditions for the differentiability of the optimal solution (z^*, p_0) with respect to $p \in P \subseteq \mathbb{R}^{N_p}$ are given in the next theorem, [8, 18].

Theorem 5.1. *Suppose $F(z, p)$ and $G^a(z, p)$ are twice differentiable with respect to z and p . And assume z^* be a strong regular local solution of (18) for a fixed parameter p_0 with Lagrange multipliers λ_0 , i.e. $G^a(z_0, p_0) = 0$ and*

1. $\nabla_z G^a(z_0, p_0)$ is full rank (z_0 is regular),
2. $\nabla_z \mathcal{L}(z_0, \lambda_0, p_0) = 0$, $\lambda_0^T G^a(z_0, p_0) = 0$ (necessary optimality conditions),
3. $\lambda_{i_a} > 0$ for $i_a = 1, \dots, N_a$ (strict complementarity),
4. $v^T \nabla_{zz}^2 \mathcal{L}(z_0, \lambda_0, p_0) v > 0$, $\forall v \in \ker(\nabla_z G^a(z_0, p_0))$, $v \neq 0$ (second order sufficient conditions),

then there exists a neighborhood \mathcal{P} of p_0 such that the problem (18) has a unique strong regular local solution $z(p)$ and $\lambda(p)$. Furthermore, $z(p)$ and $\lambda(p)$ are continuously differentiable functions of $p \in \mathcal{P}(p_0)$ and

$$\begin{pmatrix} \nabla_{zz}^2 \mathcal{L}(z_0, \lambda_0, p_0) & \nabla_z G^a(z_0, p_0)^T \\ \nabla_z G^a(z_0, p_0) & 0 \end{pmatrix} \begin{pmatrix} \frac{dz}{dp}(p_0) \\ \frac{d\lambda}{dp}(p_0) \end{pmatrix} = - \begin{pmatrix} \nabla_{zp}^2 \mathcal{L}(z_0, \lambda_0, p_0) \\ \nabla_p^2 G^a(z_0, p_0) \end{pmatrix} \quad (20)$$

where $\nabla_{zz}^2 \mathcal{L}$ denotes the Hessian of the Lagrangian.

The proof of the theorem is given in [11] and [17]. Since the coefficient matrix in (20) is non-singular by the assumption of Theorem 5.1, $\frac{dz}{dp}(p_0)$ and $\frac{d\lambda}{dp}(p_0)$ can be calculated explicitly by solving the linear equation system (20). In [11], it is explained how one can check the assumptions of Theorem 5.1 numerically by using the projected or reduced Hessian.

5.2 Real-Time Mission Correction

In this section, a mathematical method to correct the violations of the space mission trajectory is presented. This method is based on the discussion from last section about sensitivity analysis of an NLP(p), and linear approximation

$$\begin{aligned} z(p) &= z(p_0 + \Delta p) \\ &\approx \tilde{z}(p) = z(p_0) + \frac{dz}{dp}(p_0) \Delta p, \end{aligned} \quad (21)$$

which uses the explicit sensitivity differentials achieved by solving linear system (20) for perturbations Δp to modify the control signals.

To deal with the linear approximation (21), one has to consider the changes of the active constraints, see [10, 11, 6]. Although (21) results in acceptable real-time approximations for small Δp , it can cause larger deviations from the active constraints for larger Δp and leads to a non-admissible solution as

$$\varepsilon_1 := G^a(\tilde{z}(p), p) \neq 0. \quad (22)$$

Introducing an auxiliary parameter $q \in \mathbb{R}^{N_a}$ for every active constraint in (18), one deals with the following problem

$$\begin{cases} \text{minimize} & F(z, p) \\ \text{under} & G^a(z, p) - q = 0 \end{cases} \quad (23)$$

Choosing the nominal value of q which is $q_0 = 0$, the problem (23) is equivalent to the problem (18). Actually, the parameters can be considered as $(p, q) \in \mathbb{R}^{N_p + N_a}$. Since one of the problems (18) or (23) satisfies the conditions of Theorem 5.1 if the other one does, therefore one can compute the sensitivity differentials $\frac{dz}{dq}(q_0)$ and $\frac{d\lambda}{dq}(q_0)$ in the same way as (20). By using the new sensitivity differentials, we can hope that a better approximation of the form of (21) can be found to improve the optimality and admissibility of the real-time approximation. Considering (21) and (22), this approximation is given by

$$\begin{aligned} z(p) &\approx \tilde{z}_2(p) = \tilde{z}(p) - \frac{dz}{dq}(q_0)\varepsilon_1 \\ &= \tilde{z}(p) + \frac{dz}{dq}(0)G^a(\tilde{z}(p), p). \end{aligned} \quad (24)$$

Let $\tilde{z}_1(p)$ denote the same $\tilde{z}(p)$, then the improving steps (22) and (24) can be considered as an iterative process to construct sequences $(\varepsilon_k)_k$ and $(\tilde{z}_k)_k$ for $k = 1, 2, \dots$ as the parameter and solution sequences, respectively. Since the nominal solution $z(p_0)$ as well as the sensitivity differentials $\frac{dz}{dp}(p_0)$ and $\frac{dz}{dq}(q_0)$ can be computed off-line, steps like (24) do not need any derivative computational cost. Moreover, the terms of form $\frac{dz}{dq}(0)G^a(\tilde{z}_i(p), p)$, can be considered as a correcting *feedback step* for ε_i -error correction. In the following, the feedback closed loop is briefly presented. The loop continues until a prescribed accuracy ε_∞ is achieved.

1. Initialize $\tilde{z}_1(p) = z(p_0) + \frac{dz}{dp}(p_0)\Delta p$, $k = 1$ and choose the desirable accuracy ε_∞ .
2. While $\|G^a(\tilde{z}_k(p), p)\|_2 > \varepsilon_\infty$ do the following
 - $\tilde{z}_{k+1}(p) := \tilde{z}_k(p) - \frac{dz}{dq}(0)G^a(\tilde{z}_k(p), p)$,
 - $k := k + 1$.

For more details about the feedback rule and the convergence rate of $\tilde{z}_k(p)$, see [12].

References

- [1] C.J. Alpert, T.C. Hu, J.H. Huang, A.B. Kahng, and D. Karger. Prim-Dijkstra tradeoffs for improved performance-driven routing tree design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(7):890–896, 1995.
- [2] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, pages 2–11, 1996.
- [3] M. Athans and P. L. Falb. *Optimal Control*. McGraw-Hill, New York, 1966.
- [4] A. Barclay, P. E. Gill, and Rosen J. B. SQP methods and their application to numerical optimal control. In W.H. Schmidt, K. Heier, L. Bittner, and R. Bulirsch, editors, *Variational Calculus, Optimal Control and Applications*, Basel, 1998. Birkhäuser.
- [5] Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1996.
- [6] T. J. Beltracchi and G. A. Gabriele. Observations on extrapolations using parameter sensitivity derivatives. In *Advances in Design Automation*, volume 14, pages 165–174. ASME, 1988.
- [7] J. T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. SIAM, Philadelphia, PA, second edition, 2010.
- [8] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *IFAC 9th World Congress*, Budapest, Hungary, 1984.
- [9] A. Bryson and Y. Ho. *Applied Optimal Control*. Hemisphere/Wiley, 1975.
- [10] C. Büskens. *Direkt Optimierungsmethoden zur numerischen Berechnung optimaler Steuerungen*. Diploma thesis, Institut für Numerische Mathematik, Universität Münster, Münster, Germany, 1993.
- [11] C. Büskens. *Optimierungsmethoden und Sensitivitätsanalyse für optimale Steuerungsprozesse mit Steuer- und Zustands-Beschränkungen*. Dissertation, Institut für Numerische Mathematik, Universität Münster, Münster, Germany, 1998.
- [12] C. Büskens. *Echtzeitorientierung und Echtzeitorientierung parametergestörter Probleme*. Habilitation, Fachbereich Mathematik und Physik, Universität Bayreuth, Bayreuth, Germany, 2002.
- [13] Cid Carvalho De Souza and Celso Carneiro Ribeiro. Heuristics for the minimum rectilinear Steiner tree problem: new algorithms and a computational study. *Discrete Applied Mathematics*, 45(3):205–220, 1993.

- [14] Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, editors. *Column generation*, volume 5 of *GERAD 25th Anniversary Series*. Springer, New York, 2005.
- [15] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1:269–271, 1959.
- [16] Pontryagin et al. *The mathematical theory of optimal processes*. Interscience, New York, 1962.
- [17] A. V. Fiacco. *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*, volume 165 of *Mathematics in Science and Engineering*. Academic Press, New York, 1983.
- [18] R. Fletcher. *Practical Methods of Optimization*. Wiley & Sons, Chichester / New York, 1997.
- [19] M. R. Garey and D. S. Johnson. The Rectilinear Steiner Tree Problem is NP-Complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.
- [20] R. Geraerts. Planning short paths with clearance using explicit corridors. In *IEEE International Conference on Robotics and Automation*, pages 1997–2004, 2010.
- [21] P. E. Hart, N. J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
- [22] S. Held, B. Korte, D. Rautenbach, and J. Vygen. Combinatorial Optimization in VLSI design. In V. Chvátal and N. Sbihi, editors, *Combinatorial Optimization: Methods and Applications*. IOS Press, to appear.
- [23] Renato F. Hentschke, Jaganathan Narasimham, Marcelo O. Johann, and Ricardo L. Reis. Maze routing Steiner trees with effective critical sink optimization. In *Proceedings of the 2007 international symposium on Physical design, ISPD '07*, pages 135–142, New York, NY, USA, 2007. ACM.
- [24] Huibo Hou, Jiang Hu, and S.S. Sapatnekar. Non-Hanan routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(4):436–444, April 1999.
- [25] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Plenum Press, New York, 1972.
- [26] C. Y. Lee. An algorithm for path connections and its applications. *Electronic Computers, IRE Transactions on*, EC-10(3):346–365, 1961.
- [27] Christine R. Leverenz and Mirosław Truszczyński. The rectilinear Steiner tree problem: algorithms and examples using permutations of the terminal set. In *Proceedings of the 37th annual Southeast regional conference (CD-ROM)*, ACM-SE 37, New York, NY, USA, 1999. ACM.

- [28] Chih-Hung Liu, Shih-Yi Yuan, Sy-Yen Kuo, and Szu-Chi Wang. High-performance obstacle-avoiding rectilinear Steiner tree construction. *ACM Transactions on Design Automation of Electronic Systems*, 14:45:1–45:29, 2009.
- [29] Dirk Müller, Klaus Radke, and Jens Vygen. Faster min–max resource sharing in theory and practice. *Mathematical Programming Computation*, 3:1–35, 2011. 10.1007/s12532-011-0023-y.
- [30] S. Peyer, D. Rautenbach, and J. Vygen. A generalization of Dijkstra’s shortest path algorithm with applications to VLSI routing. *Journal of Discrete Algorithms*, 7:377–390, 2009.
- [31] J. Scott Provan. An approximation scheme for finding Steiner trees with obstacles. *SIAM J. Comput.*, 17(5):920–934, 1988.
- [32] Dirk Schwanenberg, Govert Verhoeven, and Luciano Raso. Nonlinear model predictive control of water resources systems in operational flood forecasting. In 55. *IWK*. TU Ilmenau, September 2010.
- [33] David Warne, Pawel Winter, and Martin Zachariasen. GeoSteiner [Computer Software]. www.diku.dk/hjemmesider/ansatte/martinz/geosteiner/. (version 3.1).
- [34] Martin Zachariasen. Rectilinear full Steiner tree generation. *Networks*, 33(2):125–143, 1999.
- [35] Hai Zhou. Efficient Steiner tree construction based on spanning graphs. In *Proceedings of the 2003 international symposium on Physical design, ISPD ’03*, pages 152–157, New York, NY, USA, 2003. ACM.

Node counting in wireless ad-hoc networks

*Joep Evers** *Demeter Kiss[†]* *Wojtek Kowalczyk[‡]*
Tejaswi Navilarekallu[‡] *Michiel Renger** *Lorenzo Sella[§]*
Vincent Timperio[§] *Adrian Viorel[¶]* *Sandra van Wijk**
Albert-Jan Yzelman^{||}

Abstract

We study wireless ad-hoc networks consisting of small microprocessors with limited memory, where the wireless communication between the processors can be highly unreliable. For this setting, we propose a number of algorithms to estimate the number of nodes in the network, and the number of direct neighbors of each node. The algorithms are simulated, allowing comparison of their performance.

KEYWORDS: wireless, unreliable network, graph, algorithm, simulation.

1 Introduction

Wireless networks are part of our everyday lives. The most commonly known wireless networks enable communication between computers or PDA's. Recently there has been an increased interest in wireless networks of smaller microprocessors with limited processing power. It is the possible large deployment of these small wireless devices, that makes them applicable to many diverse situations. By equipping nodes with sensors, networks of many such nodes can, for example, gather information on bird flocking [2], monitor social behavior and map crowd dynamics [7], and detect forest fires [12].

The technology that is currently being developed at CHES [1] enables the nodes to communicate with each other by broadcasting short messages through the air. However, not all nodes will actually receive such broadcasts. For nodes that are too far apart, the signal may be too weak to establish a link. But even for nodes that are physically close, the wireless communication may be asymmetric or temporarily failing. Moreover, the topology of the network is inherently dynamic due to the possible physical movement of the nodes. These factors add up to highly unreliable networks.

⁰Corresponding author: Michiel Renger, Eindhoven University of Technology, P.O. Box 513, 5600 MB, The Netherlands, + 31 40 247 2685, d.r.m.renger@tue.nl

*Eindhoven University of Technology, The Netherlands

[†]CWI, The Netherlands

[‡]VU University Amsterdam, The Netherlands

[§]University of Leiden, The Netherlands

[¶]Universitatea Babes-Bolyai Cluj Napoca, Romania

^{||}Utrecht University, The Netherlands

Previous experiments at CHESS have been successful in synchronizing the internal clocks of all nodes in a network with each other. To save the battery lifetime of each node, one would like to reduce as much as possible the time interval in which a node listens to incoming messages. However, a node can only receive a message from one other node at a time. Thus, a trade-off has to be made between saving battery power and receiving sufficient messages. To this aim, each node needs to have information about the size of the network it is in. The goal of this study is to design robust algorithms that retrieve this information, using minimal processing and storage memory.

The paper is organized as follows. In Section 2 we explain the problem in more detail, and state the main assumptions for this research. We describe a number of algorithms to estimate the neighborhood size in Section 3, and to estimate the network size in Section 4. Then, in Section 5 we discuss the simulation of the algorithms and the corresponding results. Finally we draw conclusions and give suggestions for further research in Section 6.

2 Problem Description

For the goal of the research presented in this paper, we propose a number of algorithms that estimate, for each node:

1. the number of direct neighbors (also called the degree), and
2. the total network size.

We will assume that the algorithms proposed in this paper are much faster than the changes that occur in network topology. This motivates our choice to consider only networks that have fixed topology, although some random noise in data exchange is allowed. Naturally, this assumption restricts the robustness of the algorithms: the longer they take, the more likely the topology will change in practice, yielding unreliable estimates.

As mentioned, each node has its own clock; the available technology allows synchronization of these clocks. However, synchronization only works between nodes within a (sub-) network. In practice, it may take some time until all nodes in a network are synchronized, especially if two separated sub-networks merge. Because of the different time scales assumption, the nodes in the networks under consideration are synchronized.

Typically, the clock of each node has a period of $1s$, which is divided into an *active period* (of the order of 1% of the total period) and an *inactive period*. During the inactive period, the node cannot receive any messages, which saves its battery life considerably. The active period is subdivided into a number of *slots* (typically multiples of 4 or 8). Within one time slot, a node can receive one message from one other node only; if messages from multiple nodes arrive at another node within the same slot, the messages will interfere, and the receiving node can no longer distinguish the signal from noise (in this case we speak of a *collision*). See Figure 1.

Each node has a unique identifier; one trivial way to estimate the neighborhood and network size would be to simply let each node broadcast its identifier together with all identifiers it has received so far. However, this method requires too much bandwidth

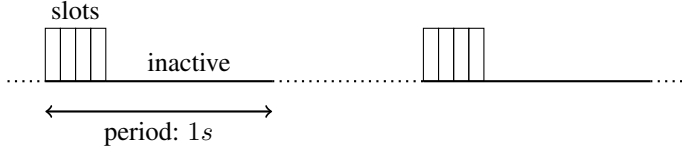


Figure 1: The periods of a node.

and local memory. Hence, we devise less demanding algorithms for estimating the neighborhood and network size.

3 Algorithms for neighborhood size estimation

In this section we study the neighborhood size (degree) of a given node. We present four estimation algorithms. The organization is the same in all cases: first we give an informal description, then we provide pseudocode for the algorithm.

Before presenting the algorithms, we should first note that the links between nodes are possibly directional. Since a node can not directly determine how many nodes receive its message (the *outgoing degree*), we focus on the neighbors from which a node can receive a message (the *incoming degree*). Sometimes we will refer to the incoming degree as the number of potential neighbors, to stress that we include neighbors whose message may not be received due to noise or collisions.

Secondly, we note that, given the number of messages received by a single node in each of the last X time periods, a simple lower and upper bound on the number of unique neighbors d can be constructed. Denote the number of received messages by m_1, m_2, \dots, m_X . As a lower bound d_L on the neighborhood size, we take the maximum of received messages in one period:

$$d_L = \max_{1 \leq j \leq X} m_j.$$

The messages in one period are certainly coming from unique neighbors. Hence, the maximum is a lower bound: in the best case scenario, all messages originate from a neighbor already included in the lower bound. As an upper bound d_U , we take the sum of all messages received. In the best case scenario for d_U , all messages originate from different neighbors. Thus:

$$\max_{1 \leq j \leq X} m_j = d_L \leq d \leq d_U = \sum_{j=1}^X m_j.$$

Note that these bounds only incorporate the neighbors from which a message was actually received during at least one of the last X periods. It might be possible that it has more neighbors from which no messages have come through the last X periods.

3.1 Random ID: neighborhood size estimation

Here we describe an algorithm similar to the basic algorithm given at the end of the Section 2. The main difference is that we use randomized identifiers in stead of the built-in ones as follows.

Let L be the range of the random IDs, a parameter of the algorithm. Each node v picks a uniform random number from $\{1, 2, \dots, L\}$, which we refer to as random identifiers and denote by RID_v . Also each node v uses R_v a bit vector of length L , set to 0 at start.

Each node in each round sends a signal containing its random ID. When a node v receives a signal from node w , then set the RID_w th bit of R_v to 1. The vector R_v stabilizes with a list of the random IDs of the neighbors of v . Then we use the estimator in (2) explained below. The pseudocode is given in Algorithm 1.

After R_v is stabilized, the probability that the first bit of R_v is equal to 0 is given by

$$P(R_v(1) = 0) = \left(1 - \frac{1}{L}\right)^d. \quad (1)$$

Replacing $P(R_v(1) = 0)$ in Eq. (1) with

$$V_v = \frac{\#\{i \mid R_v(i) = 0\}}{L},$$

the proportion of the bits with value 0 in R_v provides the estimate

$$\hat{d}_v = -\frac{\log V_v}{\log(1 - \frac{1}{L})} \approx -L \log(V_v). \quad (2)$$

To conclude, we discuss the choice of the parameter L . Using Taylor expansion, Whang et al. [15] show that, the bias and standard error of $\frac{\hat{d}_v}{d}$ satisfies

$$E\left(\frac{\hat{d}_v}{d_v} - 1\right) / \frac{e^t - t - 1}{2d_v} \rightarrow 1, \quad (3)$$

$$StdError\left(\frac{\hat{d}_v}{d_v}\right) / \frac{\sqrt{L(e^t - t - 1)}}{d_v} \rightarrow 1 \quad (4)$$

as $d_v, L \rightarrow \infty$ with $\frac{d_v}{L} \rightarrow t$.

In [15], only the case $L = \mathcal{O}(d_v)$ was considered. Hence, they keep open the possibility to use the estimator \hat{d}_v for smaller values of L . However, a simple computation based on the expected number of 0 bits of the stabilized R_v shows that we have to choose $L > c \frac{d_v}{\log d_v}$ for some constant c to be able use the estimator \hat{d}_v . Hence we have to use at least $\mathcal{O}\left(\frac{d_v}{\log d_v}\right)$ of memory to get a non-trivial estimate with this algorithm. This is not a significant improvement on the algorithm described at the end of Section 2 which uses $\mathcal{O}(d_v)$ memory. Nevertheless, we can use equations (3) and (4) to tune the parameter L for the desired accuracy, in the same way as in [15]. This results in a constant multiple (depending on the required accuracy) of memory decrease compared to the basic algorithm of Section 2.

3.2 Parameter estimation of binomial distribution

A given node has d potential neighbors, i.e. other nodes from which it might receive messages. Assume that with probability $p \in [0, 1]$ such a message from a potential

Algorithm 1 Random ID: neighborhood size estimation.**Parameters:**Range for random IDs: L

Initialise:

- 1: Set R to an array of bits with length L of 0-s.
- 2: Set RID to $rand(1, L)$.
- 3: $R(RID) = 1$

Each round:

- 1: $bsp_broadcast(RID)$
- 2: **for** $i = 1$ to bsp_nlsots **do**
- 3: $r = bsp_get(i)$ where r is the sent random ID.
- 4: Set $R(r) = 1$.
- 5: Compute V_v and $-L \log(V_v)$ – that is the estimate

neighbor is indeed received in a period, independently and identically distributed for all messages, all potential neighbors, and all periods. Then, the number of messages that this nodes receives during a period, denoted by the random variable M , follows a binomial distribution with parameters d and p .

We are given the number of messages received by the node during the last X periods. Let these numbers be given by the vector $m = (m_1, \dots, m_X)$, where each $m_j, j = 1, \dots, X$, is a realization of M . From this information, we want to estimate d , as this gives an estimate for the number of unique neighbors the node has seen during the last X periods.

So, we reduce the problem to estimating the parameters d of a binomial distribution, when X observations are given and p is unknown. An option would be to use the maximum likelihood estimator (MLE), or use the method of moments estimator (MME), however, both are known to be unstable [3]. That is, if instead of some m_j one observed $m_j + 1$, this can result in a relatively large change in the estimator for d . Therefore, we chose a more stable estimator, more precisely, the one given in DasGupta and Rubin [5]. It uses three variables that are derived from the vector m , namely the mean, denoted by \bar{m} , the maximum, m_{max} , and the sample variance:

$$S_m^2 = \left(\sum_{j=1}^X (m_j - \bar{m})^2 \right) / (X - 1).$$

Then, the estimator for d , \hat{d} , is given by (cf. [5, Eq. (8)])

$$\hat{d} = \frac{m_{max}^2 S_m^2}{\bar{m} (m_{max} - \bar{m})}. \quad (5)$$

The proposed estimator of [5] actually involves an extra parameter α , which can be used for fine tuning the estimator. We set $\alpha = 1$ here, which is claimed to be a good generic choice [5, p. 397]. A more complex estimator is proposed [5, Eq. (11)] as well, which uses a summation over inverse Beta functions. We, however, judge this

Algorithm 2 Parameter estimation of binomial distribution.

Parameters:

Number of previous periods taken into account: X

Initialise:

- 1: Let $m = (m_1, \dots, m_X)$ be the vector of received messages in each period, initialize m to be zero-vector of length X .

Each round:

- 1: Count the number of received messages in round: m_0 .
 - 2: Update m to be $m = (m_0, \dots, m_{X-1})$.
 - 3: Calculate sample mean \bar{m} , sample variance S_m^2 , and maximum m_{max} based on this m .
 - 4: Compute $\hat{d} = \frac{m_{max}^2 S_m^2}{\bar{m}(m_{max} - \bar{m})}$ (Equation (5)) - this is the estimate for number of unique neighbors.
-

one to be too complicated and time consuming to be executed on each node in every period.

Now, (5) gives an estimate for the number of (unique) neighbors of the node under consideration seen during the last X periods. In the next period, a number of messages is received by this node, say m_0 , and the vector m is updated accordingly: m_X is discarded and m_0 is prepended to m . All steps of this estimator are given in Algorithm 2.

3.3 Future broadcast

The key idea is to broadcast a message consisting of a set of random numbers which are used by the receivers to avoid repetitions in counting the number of their neighbors.

We denote by k the size in bits of the random numbers that used generated by each node, by l the size of the set of random numbers that a node broadcasts in a single message and by X the number of active periods during which one wants to estimate the number of their neighbors. In any application, there will be an intrinsic relation between these variables. In particular, l should equal the average number of broadcasts by a node in X active periods.

A broadcast message from a node consists of a list of size l , whose elements are k -bit random numbers. Once the message is broadcast, the node removes the leading random number from the list and appends a newly generated random number to the end of the list. This new list is then used in its next broadcast.

On the receiving end, a node maintains a list of *recently* seen random numbers and a list of counters whose elements keep the count of number of neighbors in the *recent* active periods. In an active period, each received broadcast message is discarded if the leading random number in the message is an element in the list of random numbers. If not, the list of random numbers in the message is appended to that of seen random numbers. A new counter is added to the second list which keeps track of the number of new neighbors seen during the current active period. At the end of the active period, both the lists are updated by removing the *old* random numbers and the *old* counter of

Algorithm 3 Future broadcast.

Parameters:Length of random numbers: k Number of active periods to be considered: X Average number of broadcasts by a node in X active periods: l **Initialise:**

- 1: Set M to an array of l random numbers between 1 and $2^k - 1$.
- 2: Set RR to an array of length X of arrays.
- 3: Set C to an array of length X .

Broadcast:

- 1: $broadcast(M)$
- 2: Remove the last element from the list M .
- 3: Insert ($rand(1, 2^k - 1)$) at the beginnngin of the list M .

Each active period:

- 1: Set $c = 0$.
- 2: Set $temp = []$.
- 3: **for** each received message **do**
- 4: **if** $M[0] \notin RR$ **then**
- 5: $append(M, temp)$.
- 6: $c = c + 1$.
- 7: Remove the last element from the list RR .
- 8: Remove the last element from the list C .
- 9: Insert $temp$ at the beginning of the list RR .
- 10: Insert c at the beginning of the list C .

Count the number of neighbors:

- 1: Return($sum(C)$).
-

number of neighbors.

The broadcast message simply consists of a list M of l k -bit random numbers. Every node maintains two lists of length X each: a list RR of lists of random numbers seen during each of the last X active periods and a list C of number of neighbors seen during each of the last X active periods. In an active period, both these lists are appended at the end with the newly seen random numbers and the number of newly seen neighbors respectively and the leading elements are deleted.

The sum of all the elements from the list C gives an estimate of the number of unique neighbors seen during the last X active periods. All steps are given in Algorithm 3.

3.4 Counting neighbors of neighbors

We now focus on the average degree of a node in the network, using this average as a (rough) estimate for the number of neighbors. For this, we want a node to count its neighbors, the neighbors of its neighbors, the neighbors of the neighbors of its neigh-

bors, and so on. Each node has to broadcast a small list containing these numbers. It also receives these lists from its neighbors, from which it updates its own list. Note that there is inevitable double-counting of nodes in this approach, but that is taken into account in this method, and hence it is not a problem.

Let $n_i^{(1)}$ be an estimate for the neighborhood size of node i , e.g. as in (5) or just the number of messages received during the last period. These are the ‘first degree’ neighbors. A message received from neighbor j includes node j in its estimate of its neighborhood size $n_j^{(1)}$. The number of neighbors of neighbors for node i is now simply the summation over these numbers, resulting in an estimate for its ‘second degree’ neighbors (all nodes at most two connections away): $n_i^{(2)} = \sum_{j \in \mathcal{N}_i} n_j^{(1)}$, where \mathcal{N}_i is the set of all neighbors of node i in the last period. As these numbers are broadcast as well, node i gets the following estimate for its ‘third degree’ neighbors: $n_i^{(3)} = \sum_{j \in \mathcal{N}_i} n_j^{(2)}$. Theoretically, we can continue this to every desired degree:

$$n_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} n_j^{(k)},$$

where $k = 1, \dots, K$ for some arbitrary K . However, the information to be broadcast by every node increases in the length of the list $(n_i^{(1)}, n_i^{(2)}, \dots, n_i^{(K)})$. Therefore, we propose to send only the list $(n_i^{(1)}, n_i^{(2)}, n_i^{(3)})$, i.e. $K = 3$, that consists of three numbers, so a node can compute up to its fourth degree neighbors $n_i^{(4)}$. In a graph that satisfies a power law for the degrees at its nodes, typically all nodes are connected within six connections. Also, an Erdős-Renyi random graph typically has a small diameter. Hence, we argue that knowledge up to the fourth degree neighbors is a good balance between knowledge of the network and the amount of data that has to be broadcast.

We use these counts to come up with an estimate for the average number of neighbors (degree) of a node in the network. We denote this average by \bar{d} . Hence, a node has on average \bar{d} neighbors, so $n_i^{(1)} \approx \bar{d}$. These \bar{d} neighbors have each \bar{d} neighbors ($n_i^{(2)} \approx \bar{d} \cdot \bar{d}$), and each of them have again \bar{d} neighbors ($n_i^{(3)} \approx \bar{d}^3$), and so on. From this a list of estimates for \bar{d} follows:

$$\left\{ \left(n_i^{(k)} \right)^{(1/k)} \right\}_{k=1}^{K+1}.$$

We expect the entries of this list to converge. We use the last element from this list in the estimator for \bar{d} , the average degree of a node. So, the estimator for \bar{d} , say \hat{d} , is given by:

$$\hat{d} = \left(n_i^{(K+1)} \right)^{1/(K+1)},$$

where we have chosen $K = 3$ here. We use \hat{d} as an estimate for the number of neighbors of the considered node. All steps of the estimator are given in Algorithm 4. Moreover, as actually the average number of neighbors of an arbitrary node is estimated, we use it as well in Algorithm 8 of Section 4.4 for estimation of the network size.

Algorithm 4 Counting neighbors of neighbors**Parameters:**

Maximum degree of neighbors taken into account: K

Initialise:

- 1: Let $n^{(k)}$ be the number of k th degree neighbors, initialize $n^{(k)} = 0$ for $k = 1, \dots, K$.

Each round:

- 1: Broadcast $(n^{(1)}, \dots, n^{(K)})$.
- 2: Set $n^{(1)}$ to estimate number of unique neighbors.
- 3: From the received vectors $(n_j^{(1)}, \dots, n_j^{(K)})$, calculate and update $n^{(1)}, n^{(2)}, \dots, n^{(K+1)}$ using $n^{(k+1)} = \sum_j n_j^{(k)}$, for $k = 1, \dots, K$.
- 4: Compute $\hat{d} = \left(n^{(K+1)}\right)^{1/(K+1)}$ - this is the estimate for the average degree of an arbitrary node in the network.
- 5: Let \hat{d} also be the estimate for number of unique neighbors.

4 Algorithms for network size estimation

In this section we present a number of algorithms for the estimation of the network size, i.e. the total number of nodes in the network. In each case, we first give an informal description of the algorithm, followed by a pseudocode. Additional to the algorithms presented in this section, we have included in the Appendix one algorithm that needs more investigation of its possible implementation and performance.

4.1 Epidemic algorithm

The idea of the method is to start a diffusion-like process such that after enough time an initially concentrated (in one node) quantity will be uniformly distributed over all nodes. If the initial quantity is exactly 1, then the amount present at each node when equilibrium is reached should be $1/N$, where N is the size of the network.

This approach is not new and we follow ideas of Jelasity and Montresor [9] who apply the anti-entropy epidemic protocol introduced by Demers et al. in their seminal work [6] to the aggregation problem. Aggregation is defined as a ‘collective name of many functions that provide global information about the system’ [9].

Jelasity and Montresor prove that using this method in a fully connected network with reliable bidirectional links, all nodes estimate N continuously and adaptively: the estimates converge exponentially fast to the exact size of the network. Furthermore, the convergence rates and memory requirements are independent of N and the same for all nodes.

More precisely, let $V(j)$ denote the amount of the aforementioned quantity, present in node j . In a perfect network the algorithm is conservative in the sense that

$$\sum_{j=1}^N V(j) = 1$$

at all times, and has the convergence property

$$V(j) \rightarrow \frac{1}{N}, \text{ for all } j = 1, \dots, N.$$

The question is if this algorithm can still be effective in an unreliable network. Because of asymmetric links undetectable information loss is possible as a node can not establish which other node has received its broadcasts. The effectiveness and accuracy of the algorithm in directional, unreliable networks will depend on the network under consideration, and can only be studied by extensive simulations.

Nevertheless, heuristically we can argue the following. If we restrict our analysis to one node, there are two ways in which its communications are affected due to the unreliability of the network:

1. some of the potentially incoming messages are lost which leads to an underestimation of the neighborhood size
2. some of the messages broadcasted by the node are also lost.

Thus, the presented algorithm may have a natural tendency to correct errors in estimating the size of the network due to 1 and 2, if the incoming error and the outgoing error would cancel themselves out.

The pseudocode of the algorithm is given in Algorithm 5.

4.2 Random ID: network size estimation

The set-up is the same as Section 3.1. The main difference from the algorithm in Section 3.1 is that now each node will not only broadcast its random ID, but also the list of random IDs which it already encountered. An additional difficulty arises, since usually the recurrence vector does not fit in the message, since the network size is usually much bigger than the length of the message which can be broadcasted in one round. Hence we divide the recurrence vector of pieces of l bits, and send the pieces separately, in random order. The algorithm is the following:

Let L be the range of the random IDs, and l is the length of the broadcasted piece, parameters of the algorithm. Then each node v picks a uniform random number from $\{1, 2, \dots, L\}$, which we refer to as random identifiers and denote by RID_v . Also each node v uses R_v , a bit vector of length L , set to 0 at start.

Each node in each round picks a uniform random number k from $\{1, 2, \dots, \lfloor L/l \rfloor\}$. It sends a signal containing RID_v , k and the k th part of its recurrence vector, which we denote by $R_{v,k}$. When the node v receives a message from w , then it sets the RID_w th bit of R_v to 1, takes $k = k_w$ and updates $R_{v,k}$ to $R_{v,k} \vee R_{w,k}$, where \vee denotes the coordinate-wise maximum. Then we use the estimator of EQ. (2). The pseudocode is given in Algorithm 6.

Note that the vector R_v stabilizes with the list of random IDs in the network for all nodes v . Using the arguments of Section 3.1 we get results similar to those in Section 3.1 for the memory usage of the algorithm.

Algorithm 5 Epidemic algorithm.

Parameters:

None

Initialise:

- 1: Assign to all nodes except one node called i the value zero. To i assign the value one:

$$\begin{aligned} V(j) &:= 0, \text{ for } j \neq i; \\ V(i) &:= 1. \end{aligned}$$

Each round:

- 1: Each node j estimates its outcoming degree by counting the number of incoming messages
- 2: Each node sends a message containing

$$m(j) = \frac{V(j)}{d_{in}(j) + 1}.$$

- 3: Each node updates its value

$$V(j) := \frac{V(j)}{d_{in}(j) + 1} + \sum_{k \text{ incoming msg}} m(k).$$

Algorithm 6 Random ID: network size estimation.

Parameters:Range for random IDs: L Length of the pieces recurrence vector: l

Initialise:

- 1: Set R_v to an array of bits with length L of 0-s.
- 2: Set RID_v to $rand(1, L)$.
- 3: $R_v(RID_v) = 1$

Each round:

- 1: Set $k = rand(1, \lceil L/l \rceil)$.
 - 2: $bsp_broadcast(RID_v, k, R_{v,k})$
 - 3: **for** $i = 1$ to bsp_nlsots **do**
 - 4: $g_1 g_2 g_3 = bsp_get(i)$ where g_1 is the sent random ID, g_2 position, g_3 is the part of the recurrence vector
 - 5: Set $R_v(g_1) = 1$.
 - 6: Set $R_{v,g_2} = \max(g_3, R_{v,g_2})$.
 - 7: Compute V_v and $-L \log(V_v)$ – that is the estimate
-

4.3 The MinTopK algorithm

Let us assume, as in the previous section, that each node keeps its private random identifier, RID_v , that is drawn uniformly from a set of identifiers that is much bigger than the network size N , so it is safe to assume that all identifiers are different. For the sake of simplicity of our presentation we will view these identifiers to be a uniform sample of size N , drawn from the interval $[0,1]$. The key idea behind our algorithm is based on an observation that the nodes can collectively create a list (shared by all the nodes) of K biggest values, where K is relatively small, for example $K = 100$, depending on the size of the available memory. The smallest element of this list, call it s , can be used to estimate the network size. Indeed, if the interval $[s, 1]$ of length $1 - s$ contains K values, then the whole interval $[0, 1]$ is expected to contain about $K/(1-s)$ values (they are uniformly distributed), so the network size can be estimated by $K/(1-s)$.

In the rest of this section we work out some details of our method. First, we present an algorithm that each node has to execute in order to create the list of top K values that are known to network nodes. Second, we provide a slightly better formula for estimating the network size from the smallest value of this list and experimentally evaluate the accuracy of our approach as a function of K and N . Finally, we make some recommendations for further research.

To create a list of the top K values we propose the following algorithm. It is well known that calculating the maximum of values that are distributed along the nodes of the network is a relatively simple task: the simplest gossip-based algorithm which asks network nodes to propagate only the biggest value they have ever seen, converges, under some assumptions, exponentially fast to a configuration, where each node knows the maximal value. This style of propagating information through the nodes was successfully generalized to compute other aggregates, such as average, variance, count, see [11] or [10]. Our algorithm finds a list of k biggest values that are distributed along the nodes using the same principle. During the initialization, each node creates a local list of K values that are initially set to 0. The only value that is known to a node - its own identifier - is stored in the list. Then the nodes start to exchange information: each node broadcasts a randomly selected non-zero value from its local list and, whenever a new value is received, it is inserted into the list as long as there is a place for it - the smallest element of the list can be deleted to make a space for the new value, as long as this new value is bigger than the smallest one. Clearly, the network will eventually converge to a state, where each node keeps a list of the top K values. The pseudocode of the final procedure for estimating the network size is presented in Algorithm 7.

To estimate the network size from MinTopK we proceed as follows. As mentioned earlier, if s denotes the smallest of the top K values then $K/(1-s)$ is a good estimate of the network size. However, it may happen that the network size is smaller than K . In such a situation the constructed list of biggest values will contain exactly N non-zero elements, so the exact network size can be found by counting these non-zero values. However, it is clear that in case $K < N$, the estimate of the network size will be only an approximation of the true node count. Instead of quantifying the quality of this approximation in an analytical way, we have estimated this error by generating 100.000 uniform samples of size N and applying our formula to the minimum of the top K elements of each sample. The results are summarized in Table 1.

Obviously, the errors reported in the table provide only a lower bound on the actual errors made by the MinTopK algorithm. In practice, depending on the network topology, reliability of connections and the number of send messages, the actual errors might be bigger: the lists that are maintained by the nodes need some time to converge to correct values. One way of estimating these errors is by running extensive simulations. Some initial results of such simulations are described in Section 5.

	K=8	K=16	K=32	K=64	K=128	K=256
N=64	31.07	18.76	10.65	0.00	0.00	0.00
N=128	32.37	20.34	12.77	7.25	0.00	0.00
N=256	33.06	20.69	13.72	8.89	5.02	0.00
N=512	33.07	21.27	14.34	9.70	6.22	3.58
N=1024	33.45	21.37	14.37	9.69	6.64	4.38
N=2048	33.12	21.57	14.51	9.97	6.94	4.67
N=4096	32.74	21.34	14.75	10.04	7.06	4.84

Table 1: The average relative absolute error (in percents) of the MinTopK estimator for various values of N and K . Each estimate is based on the results of 100.000 runs. The standard errors, all smaller than 0.4%, are not shown.

There are several research problems that could be further investigated. Firstly, we have considered only the most elementary version of the gossiping algorithm. It is obvious that this algorithm can be improved in many ways. For example, local lists could be sorted and instead of propagating only values, nodes could propagate values together with their ranks. This information could be used to speed-up the convergence, by intelligently processing the incoming information. Also, instead of selecting randomly values to be broadcasted, nodes could put more attention to broadcasting values that just entered their local lists: these seem to be most informative.

Secondly, performance of the presented algorithm should be tested on various topologies and other properties of networks. Let us note that our network might be organized in such diverse structures as a 1-dimensional grid and a fully connected graph. Clearly, the convergence speed will strongly depend on network topology.

Thirdly, it would be interesting to modify our algorithm to handle situations when nodes join or leave the network or the network topology is dynamically changing over time.

4.4 Power law

The inspiration for this algorithm was provided by a paper by R. Cohen et al. [4], in which the internet is modeled as a random graph. A network is considered in which the nodes are connected randomly to each other. The probability that a node is connected to some other node depends on the connectivity of the two. That is, the number of edges connected to the node. Connectivity is also called degree. In the paper a particular type of such random graphs is analyzed, namely those graphs in which the connectivity is distributed according to a *power law*.

More specifically, the probability that a certain node is connected to exactly k other nodes is given by:

$$P(k) = ck^{-\tau}, \quad k = m, m+1, \dots, \quad (6)$$

Algorithm 7 The MinTopK algorithm.

Parameters:Length of the list of values: K

Initialise:

- 1: create a list of K entries, each storing 0
- 2: put on the top of the list your own value
- 3: set the current estimate of network size to 1

Each round:

Broadcast:

- 1: choose at random any non-zero element from the list and broadcast it

Receive:

- 1: **for** each received identifier x **do**
- 2: find the smallest element on the list, s
- 3: **if** $s > x$ **then**
- 4: do nothing
- 5: **else**
- 6: replace s by x

Estimate network size:

- 1: find the smallest element on the list, s
 - 2: **if** $s = 0$ **then**
 - 3: set the current estimate of network size to the number of non-zero elements on the list
 - 4: **else**
 - 5: set the current estimate of network size to $K/(1 - s)$
-

where typically $2 < \tau < 3$ holds, and c is a normalization constant. m denotes the smallest possible degree (in many cases $m = 1$ will hold). For simplicity, integration is used instead of summation, such that c can be derived from the condition:

$$\int_m^\infty ck^{-\tau} dk = 1. \quad (7)$$

It follows that:

$$c = (\tau - 1)m^{\tau-1}. \quad (8)$$

A finite graph, consisting of N nodes, is considered, where the degree of each node is a random sample from the proposed power law. To estimate the maximal degree M in this graph, the assumption is made that only one node in the whole network is expected to have degree M . We denote by:

$$p := \int_M^\infty P(k) dk, \quad (9)$$

the probability that a particular node has degree greater than or equal to M . Now, the expected value of the number of nodes with degree greater than or equal to M is:

$$\begin{aligned} \mathbb{E}[\# \text{ nodes with degree } \geq M] &= \sum_{k=1}^N k \binom{N}{k} p^k (1-p)^{N-k} \\ &= Np \sum_{j=0}^{N-1} \binom{N-1}{j} p^j (1-p)^{(N-1)-j} \\ &= Np. \end{aligned} \quad (10)$$

Setting this expected value 1, we obtain:

$$\int_M^\infty P(k) dk = \frac{1}{N}. \quad (11)$$

From the above one easily derives:

$$\left(\frac{m}{M}\right)^{\tau-1} = \frac{1}{N}, \quad (12)$$

and thus the network size can be deduced from the minimal and maximal degree, together with the parameter τ :

$$N = \left(\frac{M}{m}\right)^{\tau-1}. \quad (13)$$

We would like to apply the ideas of [4] to our network of broadcasting nodes. We assume that the connectivity of our nodes can also be modeled by a power law distribution. This implies that an estimate of the network size N can be obtained, given that we have (good) estimates for m , M , and the parameter τ . These estimates are to be derived from the information a node receives from its surroundings. In particular we want to derive the value of τ from the average number of neighbors of a node in

Algorithm 8 Power law.

Initialise:

- 1: Perform the initializations of Algorithms 2 and 4.

Each round:

- 1: Receive messages from neighbors j from index set \mathcal{N} ; extract $d_{\min}^{(j)}$ and $d_{\max}^{(j)}$.

- 2: Perform Algorithms 2 and 4; extract \hat{n} and \bar{d} .

- 3: Set $m = M = \hat{n}$.

- 4: **for** $j \in \mathcal{N}$ **do**

- 5: $m = \min(m, d_{\min}^{(j)})$

- 6: $M = \max(M, d_{\max}^{(j)})$

- 7: Set $d_{\min} = m$ and $d_{\max} = M$.

- 8: Calculate $\tau = \frac{d_{\min} - 2\bar{d}}{d_{\min} - \bar{d}}$.

- 9: Calculate the estimate for the network size: $N = \left(\frac{d_{\max}}{d_{\min}}\right)^{\tau-1}$.

- 10: Broadcast message containing information desired in Algorithms 2 and 4, and d_{\min} and d_{\max} .
-

the network. First of all, note that the expected degree d of a particular node is given by (again using integration instead of summation):

$$\mathbb{E}[d] = \int_m^\infty kck^{-\tau} dk = \frac{\tau-1}{\tau-2}m, \quad (14)$$

provided that $\tau > 2$.

For each node we need an estimate of the average neighborhood size (\bar{d} , the average taken over all nodes), which can be obtained by the method described in Section 3.4. From \bar{d} we derive an estimate for the parameter τ , by assuming $\bar{d} = \mathbb{E}[d]$:

$$\tau = \frac{m - 2\bar{d}}{m - \bar{d}}. \quad (15)$$

For estimating m and M the own neighborhood size of a node (\hat{n}) is needed. Again, this quantity can be obtained from the algorithm described in Section 3.4. In order to get a clue of what the minimum and maximum degree in the network are, we compare \hat{n} to estimates of m and M which we receive from our neighbors. We make this more precise in the pseudocode for the algorithm, see Algorithm 8.

Note that this algorithm is based on the ideas presented in [4], but mathematical proofs for the applicability to our problem are lacking. For the moment we rely on simulations, as an indication of whether we are on the right track.

In case of a dynamic system, after some time, reset the system. This might be desirable, since the algorithm in its current form is incapable of capturing splitting-up of the network.

4.5 Random tours using convergence detection

This algorithm is based on a paper by L. Massoulié et al. [13], in which an estimate of the number of nodes in a peer-to-peer network is obtained using random walks. According to this algorithm, we start at a selected node, gather information on the estimated degree, randomly select a peer, and send the collected information there; the receiving node then repeats this process. By selecting random neighbors, we naturally construct a random walk, and this walk eventually returns at the initiating node, turning the random walk into a loop: this is also called a *random tour*. The information gathered at each node i in the tour is its estimate of the local degree d_i ; the factor $1/d_i$ was added to the prior information on the degree and got sent to the next stop of the tour.

Upon completion of the tour, this data is multiplied by the current estimate of the degree which directly gives an estimate of the number of nodes in the entire network. The proof of convergence relies on modeling the random walk process as a Markov chain process, and specifically it should be a reversible chain with a unique stationary distribution. More global assumptions are that the graph is connected; but this is not restrictive: if not entirely connected, it estimates the size of the connected component of the initiating node, thus corresponding with our definition of a stationary network. Another assumption made is that individual neighbors can be selected and communicated to, and that the connectivity of the network is stable (messages over edges arrive with probability 1). Both of the last two assumptions surely do not apply to our situation, and need to be attended to.

We now describe how this algorithm is modified to apply to broadcasting systems. Instead of creating random walks, we broadcast a message starting at the initiator, and assign this message a randomized identification number (ID) and a path length (initially 0). Nodes receiving this message will store this message ID together with the current path length and re-broadcast this message. When a node receives a message of which the ID is already known, and the path length is large enough¹, two random broadcasts have converged and can be considered to have followed a specific path; the concatenation of these two paths then form a tour, and an estimate of the network size can thus be calculated (the degree information and tour lengths can be added to each other). Due to the network unreliability, however, there is no theoretical guarantee that the obtained estimate can be expected to be close to the network size when two tours meet. As such, initial simulations should be made to determine whether this method works as is or requires further adaptation. Algorithm 9 details this algorithm in pseudocode.

4.6 Message based counting

The main problem with broadcasting all node IDs in a network and keeping a list of all received IDs, is the memory usage involved with such a scheme. The idea described here attempts to save memory usage, while in principle still being able to obtain an exact network count (again assuming a static network topology). This is done by ID compression by grouping nodes in unique network *trails*, where a trail visits nodes in a network by traversing on arbitrary edges using each edge at most once. Note that

¹this is a parameter to be experimented with

Algorithm 9 Random tours using convergence detection.

Parameters:

Minimum message ID length m_{ID} bits,
 minimum tour length m_l ,
 a probability p_{init} of initiating a counting broadcast.

Main algorithm:

```

1: Set  $d$  to the estimated neighborhood size
2: if  $rand() < p_{init}$  then
3:   Set  $r_1 \dots r_{m_{ID}}$  to a random binary number
4:   Set  $r_{m_{ID}+1} \dots r_{m_{ID}+1+\log_2 m_l}$  to the decimal number  $m_l$ 
5:   Set the remainder of bits (up to  $r_{27.8}$ ) to  $1/d$ 
6:    $bsp\_broadcast(r_1 \dots r_{27.8})$ 
7: Set  $\mathcal{M}$  to  $\emptyset$ 
8: Set  $\mathcal{C}$  to  $\emptyset$ 
9: Allocate an array  $x = x_1 \dots x_{bsp\_nslots()}$ 
10: Allocate an array  $l = l_1 \dots l_{bsp\_nslots()}$ 
11: for  $l = 1$  to  $bsp\_nslots()$  do
12:    $g = g_1 \dots g_{27.8} = bsp\_get(l)$ 
13:   if  $g$  is a valid message then
14:     Set  $i = g_1 \dots g_{m_{ID}}$ 
15:     Extract path length  $l$  from  $g_{m_{ID}+1} \dots g_{m_{ID}+1+\log_2 m_l}$ 
16:     Extract a floating point number  $f$  from  $g_{m+1} \dots g_{27.8}$ 
17:     if  $i \in \mathcal{M}$  then
18:       if  $l = 0$  then
19:         Convergence: obtain a network size estimate of  $d(x_i + f)$ 
20:         Store  $i$  into  $\mathcal{C}$ 
21:       else
22:         if  $l < l_i$  then
23:           Store  $l$  into  $l_i$ 
24:           Store  $f$  into  $x_i$ 
25:       else
26:         Store  $i$  in  $\mathcal{M}$ 
27:         Store  $f$  into  $x_i$ 
28:         Store  $l$  into  $l_i$ 
29: Select a random  $i$  in  $\mathcal{M}$  s.t.  $i \notin \mathcal{C}$ 
30: Construct a message  $g$  with the first bits containing  $i$ , followed by  $l - 1$  in
     $\log_2 m_l$  bits, and with the remaining bits set to  $x_i + 1/d$ 
31:  $bsp\_broadcast(g)$ 

```

Algorithm 10 Message based counting.

Initialise:

- 1: Set \mathcal{M} to \emptyset
- 2: Set $s = 0$

Main algorithm:

- 1: **if** $bsp_pid()$ is an algorithm initiator **then**
- 2: $bsp_broadcast((0, \{bsp_pid()\}))$
- 3: Set $s = 1$
- 4: **for** $l = 1$ to $bsp_nslots()$ **do**
- 5: **if** $g = (o, T)$ is a valid message **then**
- 6: **if** $s = 0$ **then**
- 7: **if** $|T| = M$ **then**
- 8: $bsp_broadcast(o + 1, \{bsp_pid()\})$
- 9: **else**
- 10: $bsp_broadcast(o, T \cup \{bsp_pid()\})$
- 11: Set $s = 1$
- 12: Add the sent message to \mathcal{M}
- 13: **else**
- 14: **if** $g \notin \mathcal{M}$ **then**
- 15: Add g to \mathcal{M}
- 16: **if** no message is sent yet **then**
- 17: Select b from \mathcal{M} randomly
- 18: $bsp_broadcast(b)$

nodes can thus be visited more than once; a trail thus differs from a path in this sense.

The algorithm works by sending messages of the following type: (o, T) with $0 \leq o < M - 1$ an integer and T a collection of at most $M - 1$ IDs; these messages thus have a size of M integers. Each node furthermore can store received messages in a collection \mathcal{M} , and has a local boolean s keeping track whether it has modified an incoming message. We assume the existence of an initiator, which starts with broadcasting a message $(0, \{bsp_pid()\})$, storing this message in its local \mathcal{M} , and settings its s to `true`. Other nodes start with $\mathcal{M} = \emptyset$ and s set to `false`.

Each node, when a message is received and s is `false`, adds its ID to T , sets s to `true`, and re-broadcasts. It can happen that T becomes too large after addition of the local ID; in this case, the other, older, IDs are first removed from T , and o is incremented by one. The message variable o thus counts the number of *overflows*; while the message ID, now set to the node at which it overflows, remains unique; this is by virtue of the local s variable which prevents adding a node to a message more than once. Thus the message size is compressed: the re-broadcasted message still contains enough information to obtain the number of nodes its trail visited.

If a message is received but s is set to `true` already, if it is not already in \mathcal{M} , it is added to \mathcal{M} ; otherwise it is not added. In all cases, a random entry of \mathcal{M} is re-broadcasted so that in time, nodes in the network may converge to having the same set of trails. Algorithm 10 shows an implementation of this scheme.

5 Simulation and results

In this section we first describe the graphs for which we simulated the algorithms, then we describe how we have simulated the algorithms, and finally present the results of the simulations.

5.1 Random graphs

The graphs that we use in the simulation are generated such as to mimic the conditions of sensor node networks in practice. They depend on three parameters: the given network size N , the (expected) average degree \bar{d} , and an additional probability α (which we fix as 0.8) that a directed edge will be created. Although only the topological properties of the network are relevant, these are often related to geometric properties. Therefore, we first choose random positions for each node within a two-dimensional box of size 1. Next, we calculate the radius R that models the distance that messages may cover, such that the expected degree will be as given:

$$\bar{d} = \alpha(N - 1)\pi R^2.$$

Between each pair of nodes that are within radius R of each other, a directed edge then is created with probability α . Moreover, we add a small number of completely random directed links between any two nodes with probability N/\bar{d} . These are added because experiments at CHESS have occasionally shown some unexpected links between nodes that were geographically far apart. Finally, we add to each directed edge a (uniformly) random number in $[0.4, 0.8]$ that models a probability that will be received.

We remark that the expected average degree will deviate slightly from the requested degree because of the latter random added links, and because we do not take into account that nodes may be close to the border of the box.

5.2 Simulation

A natural way to simulate the nodes is by a Bulk Synchronous Parallel (BSP) model [8, 14]. In BSP, each iteration cycle consists of two steps. During the first step (also called a superstep), each nodes performs only local calculation and processing, i.e. without communicating with the other nodes. In the second step (also called synchronization), communications starts: all nodes broadcast and their receive messages.

We simulate the dynamic topology as follows. Messages can only be transmitted along directed edges of the graph, but have a edge-specific reception probability that is fixed in advance. The order in which the nodes broadcast their messages is random; but when a node has received 32 messages, all new messages will be ignored. Previous experiments at CHESS show that this a realistic way to model the time slots and possible collision of messages in a slot.

5.3 Results

All algorithms were simulated for several given network sizes and average degrees, yielding estimates for the actual values of the network and neighborhood size in each

node. From these estimate we calculated the absolute errors, normalized by the actual values, and averaged over all nodes in the network. The results are shown in Table 2.

Some remarks about these results need to be made. First of all, most algorithms for network size estimation make use of an estimate of the (average or local) neighborhood size. The Power Law algorithm uses the Neighbors-of-Neighbors estimator for the average neighborhood size; the Epidemic algorithm uses an explicit (deliberately undershooting) estimate for the outgoing degree. For the other algorithms that needed such estimates, we used the exact values, possibly leading to better results.

Secondly, most algorithms have one or more specific parameters to improve the convergence and precision. We did not study what the optimal values for such parameters are, and choose constant values while running the simulations on different networks sizes and average degrees. Most of these parameters are related to the values that are estimated by the algorithms, which are of course unknown beforehand.

Thirdly, some algorithms like the Power Law and the Parameter Estimation algorithms assume various things about the network under consideration or the occurrences of message loss. The resulting estimates are expected to be more accurate for networks that precisely meet these assumptions.

We paid a special attention to the MinTopK algorithm that seems to be very promising. As discussed in subsection 4.3, the accuracy of this algorithm strongly depends on the value of K , see Table 1. By playing with this value one can control the trade-off between the memory used by network nodes and the accuracy of the algorithm. In our experiments, we used three values of K : 8, 32, 128. For each of these values we performed 10 simulations and averaged the results, see Table 2. The results are consistent with our expectations and are close to the estimates that are provided in Table 1. The observed deviations can be attributed to the fact that not all simulation runs converged to the “correct” configuration—this is certainly the case for networks with 1000 nodes and the number of iterations limited to 30.

The Random Tours algorithm did not yield any estimates, which is denoted in the table by a dash. This is due to the fact that, in order to get a good random tour, the minimal path length must be high enough; but due to message loss, the higher this parameter, the lower the probability that two broadcasts meet, resulting in no estimates. Moreover, even if a random tour would be completed and estimates were calculated, we expect these to be inaccurate since the assumptions of the original algorithms for bidirectional graphs are not met: the Markov process corresponding to one of the random paths generated is not necessarily reversible, recurrent, nor does it necessarily have a stationary distribution.

We observed that the Epidemic algorithm sometimes diverges, especially for larger networks with a higher connectivity. This results from too much mass being lost by failing links, such that the estimates for the network size in each node (taken as the inverse of the mass in that node) will blow up.

In general, we can see that all algorithms become more inaccurate for larger networks and increasing average degree. This is expected since in larger networks, algorithms must run longer before reaching consensus, and depending on the algorithm, too much information may become lost in the process.

	size av. degr.	100 10	100 30	1000 30	1000 100
algorithm	steps	neighborhood size estimation			
Rand. ID nbh.	30	1.5%	4.1%	16.1%	50.9%
	300	1.5%	4.1%	58.5%	59.6%
Parameter est.	30	55.3%	178.0%	252.2%	942.3%
	300	15.6%	20.7%	30.0%	95.6%
Future bc.	30	119.1%	138.5%	136.9%	33.9%
	300	116.2%	127.0%	136.6%	39.6%
algorithm	steps	average neighb. size estimation			
Counting n. of n.	30	29.9%	25.9%	27.5%	53.3%
	300	29.5%	25.0%	27.4%	53.0%
algorithm	steps	network size estimation			
Epidemic	30	76.0%	91.3%	80.0%	89.0%
	300	72.8%	91.5%	93.5%	94.0%
Rand. ID nw.	30	85.5%	55.4%	90.5%	85.4%
	300	37.1%	2.4%	5.5%	84.5%
MinTop8	30	22.7%	33.9%	23.3%	77.6%
	300	28.3%	30.4%	22.8%	39.8%
MinTop32	30	17.2%	14.0%	12.5%	72.9%
	300	11.3%	10.1%	17.4%	49.9%
MinTop128	30	0.3%	0.3%	8.8%	84.5%
	300	1.0%	0.5%	5.4%	36.5%
Power law	30	1550.2%	567.5%	98.6%	94.5%
	300	82.8%	8190.6%	98.6%	94.8%
Random tours	30	-	-	-	-
	300	-	-	-	-
Msg.b. count.	30	30.1%	22.8%	84.9%	91.7%
	300	36.4%	55.8%	77.4%	90.0%

Table 2: Simulation results: mean relative absolute errors.

6 Conclusion and Further Research

Although most algorithms did converge to a stable estimate, the relative errors were still significant, especially for larger networks. As mentioned, large network sizes are generally difficult to estimate in unreliable networks. The results of our simulations seem to suggest that the MinTopK algorithm is the most promising. However, a fair comparison between the algorithms is difficult because of the trade-off between memory and accuracy, and because all algorithms use different parameters. More research needs to be done to find smart and adaptive choices for the parameters of the algorithms, which is expected to improve the estimates considerably. Such results can be used in more extensive simulations to decide which of the algorithms are the fastest, robust and the most accurate.

Acknowledgments

We would like to thank Bert Bos and Daniela Gavidia from CHES, The Netherlands, for useful discussions during the week. Furthermore, we would like to thank Sander Dommers (Eindhoven University of Technology) for sharing his knowledge of power law graphs and Adrian Muntean (Eindhoven University of Technology) for his help during the early stages of this research.

A Appendix: Halving algorithm

In this section we describe the strategy of an algorithm which computes the exact size of a network. This strategy can be applied to networks with both static and dynamic topologies. The main idea of this algorithm is to select one half of the population of the network a consecutive number of times, at the end the nodes which represent the remainder of the halving at some stage represent some digit 1 of the binary expression of the number of node in the network.

The core of the algorithm consists of a loop with three steps:

1. in the first step a node send messages to find a partner and check incoming messages to find pairing requests of other nodes,
2. in the second step (when a node either has received a pairing request or his pairing requests has been accepted by another node) the node decides whether itself or the partner is going to be included in the selected half of the population (upgrade). The level of the upgraded node is increased by one (the initial levels of all nodes are zero),
3. if a node can not find a partner considers itself a digit (at the position of its level) of the binary expression of the size of the network and spreads this information around

The algorithm eventually converges to the size of the network, nevertheless at any time it is undecidable if the current estimation of the size of the network is the final one (except when the estimation is less than the number of neighbors) this because the

problem is semicomputable. There are a number of issues to take in account in order to ensure that the algorithm will converge.

First of all the way a node finds a partner needs to be implemented in a way which does not generate deadlocks. The most simple implementation is to set states for each stage of the pairing process, one for the following situations:

1. a node looking for a partner
2. a node who accepted the request of pairing of another node and he is waiting for confirmation
3. a paired node

To ensure the absence of deadlocks the second state needs to be split in two states, according to the role which a node has during the pair process. The role can be the one of asking the pairing, or the one of being asked the pairing, for instance the role can be decided according who has the greatest id. The same criteria could also decide which of the two nodes is going to be upgraded.

The pairing request is communicated by messages. A node is more likely to get paired with a neighbor. Nevertheless it may happen that all neighbors are paired and the only node available is distant and can not be reached by the messages because their lifespan is too short.

To cope with this issue every time a node needs to wait too long to find a partner doubles the life of his messages.

References

- [1] CHESS. <http://www.chess.nl>.
- [2] J.A. Carrillo, M. Fornasier, J. Rosado, and G. Toscani. Asymptotic flocking dynamics for the kinetic Cucker–Smale model. *SIAM Journal on Mathematical Analysis*, 42(1):218–236, 2010.
- [3] R.J. Carroll and F. Lombard. A note on N estimators for the binomial distribution. *Journal of the American Statistical Association*, 80(390):423–426, 1985.
- [4] R. Cohen, K. Erez, D. ben Avraham, and S. Havlin. Resilience of the internet to random breakdowns. *Physical Review Letters*, 85(21):4626–4628, Nov 2000.
- [5] A. DasGupta and H. Rubin. Estimation of binomial parameters when both n , p are unknown. *Journal of Statistical Planning and Inference*, 130(1-2):391–404, 2005.
- [6] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database management. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, PODC 87, pages 1–12. ACM, 1987.
- [7] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, May 1995.

- [8] J.M.D. Hill, B. McColl, D.C. Stefanescu, M.W. Goudreau, K. Lang, S.B. Rao, T. Suel, T. Tsantilas, and R.H. Bisseling. BSPlib: The BSP programming library. *Parallel Computing*, 24(14):1947–1980, 1998.
- [9] M. Jelasity and A. Montresor. Epidemic-style proactive aggregation in large overlay networks. In *Proceedings of the 24th International Conference on Distributed Computing Systems*, ICDCS '04, 2004.
- [10] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems*, 23:219–252, 2005.
- [11] D. Kempe, A. Dobra, and J. Gehrke. Gossip-Based Computation of Aggregate Information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 482–491. IEEE Computer Society, 2003.
- [12] P. Korteweg, M. Nuyens, R.H. Bisseling, T. Coenen, H. van den Esker, B.J. Frenk, R. de Haan, B. Heydenreich, R.W. van der Hofstad, J.C.H.W. in 't Panhuis, L. Spanjers, and M.H. van Wieren. Math saves the forest: Analysis and optimization of message delivery in wireless sensor networks. In *Proceedings of the 55th European Study Group Mathematics with Industry*, pages 117–140. Technische Universiteit Eindhoven, Jan-Febr 2006.
- [13] L. Massoulié, E. Le Merrer, A. Kermarrec, and A. Ganesh. Peer counting and sampling in overlay networks: random walk methods. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, PODC '06, pages 123–132, New York, NY, USA, 2006. ACM.
- [14] L.G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33:103–111, August 1990.
- [15] Kyu-Young Whang, Brad T. Vander-Zanden, and Howard M. Taylor. A linear-time probabilistic counting algorithm for database applications. *ACM Transactions on Database Systems*, 15(2):208–229, June 1990.

Strategic bidding in a primary reserve auction

*Denes Palvolgyi** *Gergely Csapo** *Meike Wortel†*
Thijs Ruijgrok‡ *Wander Wadman,§* *Zsombor Meder**

1 Introduction

Electricity grids are subject to a constant change of demand. If a power line is overloaded, the demand is rerouted to another line, which is then also likely to overload due to the sudden spike in voltage. Due to this cascading effect a grid-wide blackout is not at all improbable; one occurred in Italy in 2003. The costs of such a blackout are immense in today's modern society. Transport and telecommunication systems have such a high power demand that a backup power generator system would come at a very high cost. To solve this, Germany requires the electricity grid operators to have Primary Reserve Capacity on standby. Primary Reserve Capacity is provided by power plants that are able to go online and start generating power at 10 seconds notice. Such a feat is technically complicated, and providing plants require government certification. In Germany the PRC is obtained at a monthly auction: the bidders submit pairs of quantity and price/unit ratios: they are willing to provide a capacity of x Megawatts at a price of y euros/Megawatt. A supplier may divide his capacity and submit multiple bids, but he has to be able to provide the summed capacity of his accepted bids. The demand of the grid operators is predetermined by past statistics, and the demand is perfectly inelastic due to legal reasons. The auction mechanism is such that capacities with the lowest price/unit ratios are bought until the demand is fulfilled. If the quantity of the last bundle bought exceeds the remaining demand, only the necessary part of the quantity is bought. The auction results are made publicly available, but only the accepted offers. Therefore bidders cannot easily estimate the strategies of the other bidders. However PRC-capable power plants are expensive. The players in the market have good estimates of each others' capacities. Such a market - known demand, few suppliers with approximately known capacities and algorithmic market mechanism - seems ideal for economic analysis. However, it is unclear whether the human decision makers involved introduce too much chaos into this system. In section 2 we present the game theoretical model that has been solved by Kreps and Scheinkman [2]. In section 3 we give up the assumption of rationality and examine models of bounded rationality. In section 4 we discuss the results of econometric models run on time series.

*Maastricht University

†VU University Amsterdam

‡Utrecht University

§Technical University Delft

2 Game theoretical model

The Nash-equilibrium solution of a more complicated case is already provided in Kreps and Scheinkman [2]. They discuss the problem of a Bertrand duopoly¹. with capacity constraints and zero production costs, but their findings are easy to extend. Let us assume that there are n players and the capacity of player i is $x_i > 0$. First let us assume that the 'production' or activating capacity is 0. Let D denote the demand. If there is no j such that

$$\sum_{i=1}^n x_i < D + x_j,$$

then the competitive equilibrium of this game is that all firms offer their capacities at 0 price. In a situation when firm were offering their capacities at a positive price there is unavoidably someone who cannot sell his complete capacity. This firm can gain by selling his leftover capacity at half the price, so there is no other equilibrium.

Assume that the activation of capacities is costly, or there is an alternate market for the leftover capacities (which is true in our case: PRC plants can also produce energy at constant rate for long durations) and the per unit costs of the firms are such that $c_1 \leq c_2 \leq \dots \leq c_n$. Let j be the smallest number with

$$\sum_{i=1}^j x_i > D.$$

Then the equilibrium price is c_{j+1} . In situation with price $p > c_{j+1}$ there would again be a firm with unsold capacity and he could gain by selling it for $\frac{p+c_{j+1}}{2}$.

If there is a firm j such that

$$\sum_{i=1}^n x_i < D + x_j,$$

then we run into technical difficulties. Without the capacities of firm j , the demand cannot be satisfied, and demand is inelastic. So firm j can theoretically ask for *any* price, so there is no optimum price and hence no equilibrium. Obviously this is not the case in the real world. Such behaviour would attract investigations by the authorities for abusing market power. Assuming there is a highest price which is not yet suspicious, the firm j has to assess whether it is best to sell his partial capacity at the maximum price or his full capacity at competitive price. If he benefits more by selling his partial capacity at maximum price then there is no pure equilibrium as the other firms have no incentive to sell at a much lower price, and a price war of infinitesimal undercutting would ensue. For the mixed equilibrium see Kreps and Scheinkman (1983).

It is worth noting that firms also covertly or tacitly cooperate. Assume there are just two firms, with $x_1, x_2 > D$ and $c_1 = c_2 = 0$. Let the maximum price be normalized to 1. We assume that if the firms bid at the same price the demand is distributed among them equally. As we stated before the Nash-equilibrium would be

¹A Bertrand duopoly is a market with two suppliers where buyers buy from the supplier with the lowest price until his supply is used up or the demand is fulfilled.

to sell at 0 price. If both firms were to sell at the price of 1 they would both benefit. But this cooperation is not an equilibrium in the one-shot game as a firm can gain by selling at $1 - \epsilon$ if ϵ is a small enough positive number. But if the game is repeated over infinite time periods and future money is discounted at a rate of $\frac{1}{1+r}$ then there may be infinitely many equilibria. A simple one is constructed using the following strategy: I will price at 1 until the other firm does so as well. If he undercuts me I will start pricing at 0 and will do that forever. This strategy is an equilibrium if undercutting the other is not beneficial, that is there is no $\epsilon > 0$ such that

$$D(1 - \epsilon) > \frac{D}{2} \left(\sum_{i=0}^{\infty} \frac{1}{(1+r)^i} \right).$$

This is true if

$$r \leq 1$$

which is a reasonable assumption for the interest rate. If there are n firms, the equation changes to

$$D(1 - \epsilon) > \frac{D}{n} \left(\sum_{i=0}^{\infty} \frac{1}{(1+r)^i} \right)$$

and

$$r \leq \frac{1}{n-1}$$

so cooperation in equilibrium is less likely. The actual data supports this theoretical prediction when the bi-annual auction got replaced by a monthly auction. The effective interest rate dropped, and prices raised by close to 50%. While this can be interpreted as a sign of tacit collusion, it can also mean that the firms had less to lose so they were willing to risk higher bids, and this started an upward trend which encouraged further increasing bid prices.

There were regular price fluctuations on the market, which does not happen in our theoretical model, not even if we consider mixed equilibria. Therefore we decided to give up the assumption of profit maximizing behavior. These models are presented in the next section.

3 Models of bounded rationality

In this section we assume that the firms make decisions based on some simple but non-optimal rule. Of course it is impossible to guess what specific rules firms are using, so we decided to invent a few sensible ones and run numerical simulations to see what kind of dynamics evolve.

The model simulates the primary reserves market and uses the same rules for the acceptance of the bids. We simulate some players that have a certain capacity and follow certain strategies for their bids (Table 1). For the strategies the players can use the information of the accepted bids from the previous auctions and their own previous bids. We want to simulate the fact that human players have a random factor to their bidding. Hence, we are interested in the influence of small deviations from the strategies. Therefore we add a factor (R) multiplied with a random real number

Ave	Bids the average of accepted bids of the last auction for its total capacity.
Max	Bids the highest of the accepted bids of the last auction for its total capacity.
Avemax	Bids the average of the maximum and the average of the accepted bids of the last auction for its total capacity.
Updown	Always bids total capacity; If the last bid was completely accepted, bids last bid + 5, else bids the lowest bid of the last auction.
Sacupdown	Always bids total capacity; If (part of) the last bid was accepted, bids last bid, else bids the lowest bid of the last auction.
Expol	Extrapolates the highest accepted bid from the last 2 auctions and bids its total capacity on the highest expected accepted bid.
Spread	Splits its capacity in 5 equal parts and bids them with a difference of 10 between the bids; If all the bids are accepted, it replaces the lowest bid with a bid 10 higher than the highest bid last time; If at least 3 bids are declined, it replaces the highest bid with a bid 10 lower than the lowest bid it bid last time; otherwise it will bid the same.

Table 1: Strategies

$\epsilon \in [-1, 1]$ to the bids. We simulated 2000 rounds of bidding for a total demand of 80 with 7 players specified in Table 2. We added a random value between -1 and 1 to all bids ($R = 1$).

The maximum accepted bid shows a slow decline (Fig. 1). If we do the simulations without adding a random factor the bidding will converge (results not shown).

If we look at the relative payoffs of the different strategies (the money they made divided by their capacity) we see that Ave and Avemax are doing best (Fig. 2). Sacupdown and Updown are doing almost as well, there is only a difference in payoffs in the beginning (influenced by their initial bid). These are the strategies that do not aim to bet the maximum and always get accepted. Player 6 playing Max had a high capacity and suffered from only getting accepted partially.

Player nr	Capacity	Strategy	First bid (Price, Capacity)
1	15	Sacupdown	(710, 15)
2	5	Ave	(690, 5)
3	15	Avemax	(700, 15)
4	5	Expol	(700, 5)
5	10	Spread	(680, 2) (690, 2) (700, 2) (710, 2) (720, 2)
6	40	Max	(700, 40)
7	30	Updown	(610, 30)

Table 2: Capacity and strategy of the players

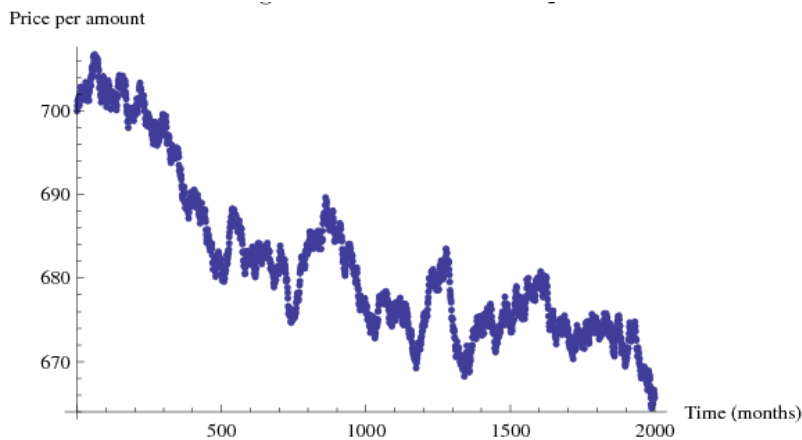


Figure 1: Maximum accepted bid over time

Including some random factor in the bidding will decrease the accepted bids in the auction over time. This is because random lower bids are always seen by the other firms, while random higher bids are sometimes not accepted and therefore not seen by the other firms. Since firms base their strategies on the bids of the last auction they will have a more negative view of the auction if there is a bigger random factor. If all firms follow the Avemax strategy and there is no random factor the market price will stabilize. However, with a random factor the price will decrease (Fig. 3).

We show that the fact that there is some random spread in the bidding will lower the bids in the auction if the firms use a strategy based on the accepted bids of the last auction. In our simulation the strategies that were not too optimistic are doing best. However, all the strategies here are fairly optimistic, if we add strategies that aim for just being accepted (for example bid the minimum bid of the last time), the price would drop very fast. For example, if the sixth firm, playing Max, decided to play Min instead, the price would drop to close to zero within 85 rounds of bidding. Therefore, even though Max was not doing too well, it is doing better than a strategy Min would do in its place. Since firms are looking to maximize their profit they are interested in their payoff and not their relative payoff compared to other firms. So overbidding or withholding capacity might be a good long term strategy. If the capacity is a lot higher, the price will drop fast and if the demand is close to the total capacity the price might even increase. This could justify holding back some capacity because the demand will then be close to the total capacity (or offer at a high price, which is effectively the same, because the other firms won't see the bid). Past data from the German PRC auctions confirms this, see the huge price increases in November 2008 and April 2011. We don't take the value of the unused capacity into account here. That would also imply a minimum bid by all firms and it could be interesting to include, especially if this value is not the same for all firms. It might also be interesting to include strategies that use their capacity relative to the total demand into account.

We have shown a specific example of a few strategies. However, this shows some general trends such as: the decline caused by random fluctuations, the importance of the total capacity relative to the demand and the effect of a optimistic or pessimistic

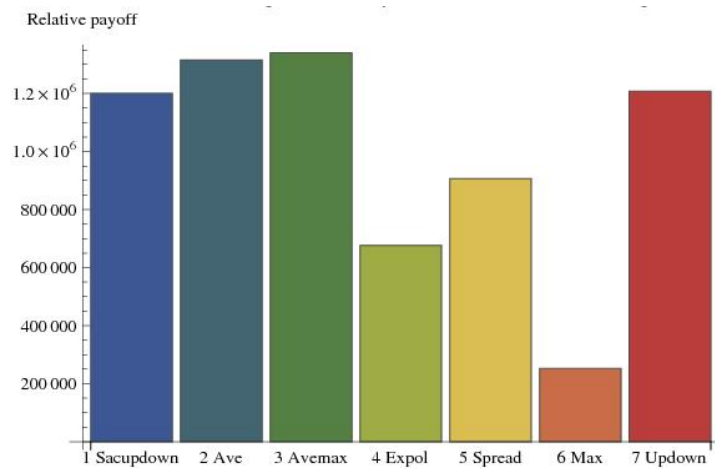


Figure 2: Payoffs of different strategies

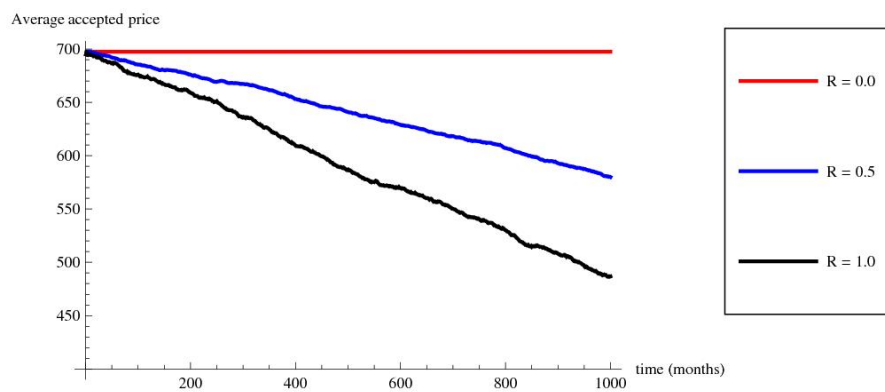


Figure 3: Effect of random changes in the bid if all firms play Avemax.

strategy on the general trend of the auction. To generalize these results or use them for a specific case the strategies have to be expanded or calibrated.

4 Time series analysis

A natural question was whether time series analysis could yield any promising strategy. Adopting the general attitude of econometric modeling, we assumed that nothing fundamental is known about the data-generating process, therefore our available dataset comprised just the set of accepted bids – pairs of quantities and prices – from December 2007 to January 2011. The question was whether using this data we could find any algorithm that could generate either expectations for the next months, or even a specific bidding strategy that would outperform experts.

First, it was necessary to trim the dataset. Since there was evidence that the data

for the first thirteen months was unreliable for prediction purposes, market conditions having changed vastly within this period – e.g. there were a different number of companies involved, consequently, a smaller overall supply for the primary reserves – we built models using only data starting with January 2009. Notwithstanding our reservations about the length of the resulting time series (just 25 time periods), we attempted to uncover the autoregressive moving average (ARMA) structure of the process, checking whether it could be used for prediction purposes. The ARMA model is a statistical tool that helps to understand the development of a process in time. The model tries to approximate the next point of the time series by taking into account the previous observations and the error terms. For further explanation consult to [1, pp. 43-72]. The advantage of autoregressive models is precisely that they require no theoretical assumptions about the *causal* background of the time series in question, and we thought that this was fitting to our purposes.

The problem of condensing the dataset of observation pairs into a more manageable time series also arose. Consequently, we decided to concentrate on prices, and only take quantities into account as weights for the average price, the latter having obvious implications for bidding policy. For the average only actually accepted quantities were included.

Autoregressive analysis first required us to check whether the series contains unit roots, and to how many levels. For the most common significance level of 5%, the examined process was integrated once. Therefore, the ARMA model was written for the (once) differentiated variable. We have checked for autoregressive and moving average components of a maximum length of 3, giving a total of 16 possible models. The model selection criteria employed was the Schwarz criterion, which penalizes more for model complexity than the Akaike criterion. With this method, we have found that the most fitting model had an MA(3) structure, providing a promising R-squared of 35%, which was relatively robust for changing sample size, popping out as the best model in over half of the cases, including the full sample.

However, despite its relative advantage over alternative ARMA models, the forecasts of an MA(3) regression based on the difference of average price cannot be directly adapted as a bidding strategy. First of all the number of observations is obviously too small, so there is likely an overfitting on data. Moreover, this model cannot be easily interpreted economically: the terms of the equation have both negative and positive signs, and the estimated impact increases with the length of lag, which is counterintuitive. Even if we avoid a specific interpretation, forecasting out-of-data with this model sometimes suggests an average price too close to the actual maximum, which would imply that at least a part of the bid for the primary reserve would not be accepted, potentially leading to great financial losses. Finally, the generated confidence intervals are sometimes wider than the actual spread of prices, indicating that this approach is indeed unreliable.

Although other approaches were tried (including but not limited to changing the model selection criterion, generating different percentiles for the price from the original data and trying to forecast the maximum or the minimum price) econometric analysis could provide no more than *ad hoc* solutions for the specific dataset. These are consequently unusable for out-of-data forecasting. The lack of economic insight even for the best models signaled the theoretical and practical weakness of the method. This suggests that even the most successful model would have been outperformed by

actual expert decisions.

5 Conclusion

Our models could explain some phenomena in the PRC auction: the price increase when the time period changed from bi-annual to monthly is due to decreased risk. Occasional price spikes can be explained by the overbidding of a firm with significant market share. While it loses money in the short run, the slowly vanishing price increase might compensate him in the long run. However qualitative explanations of these dynamics do not enable us to give an accurate prediction of tomorrow's prices. Since a few firms have significant market share their bids have a big effect on the price. From data mining we determined that some of the decision makers at these firms use ad-hoc rules when they make their bids, hence foreseeing the equilibrium price perfectly would fall more in the realm of psychology. We advise the bidders to stick close to the minimum price of the previous month. This strategy would have resulted in acceptance of the bid in every month, and it does not give a much lower revenue than a frequently but not always accepted maximum bid would. Prices will fall in the long run, and the cost of keeping them up by withdrawn capacities is too high.

References

- [1] James D. Hamilton. *Time Series Analysis*. Princeton University Press, 1994.
- [2] D. M. Kreps and J. E. Scheinkman. Quantity precommitment and Bertrand competition yield Cournot outcomes. *The Bell Journal of Economics*, 14(2):326–337, 1983.

Analysis of a Model for Ship Maneuvering

M. Apri^{*} *N. Banagaay*[†] *J.B. van den Berg*[‡] *R. Brussee*[§]
D. Bourne[†] *T. Fatima*[†] *F. Irzal*[†] *J. Rademacher*[¶]
B. Rink[‡] *F. Veerman*^{||} *S. Verpoort*^{**}

Abstract

We analyze numerically and theoretically steady states and bifurcations in a model for ship maneuvering provided by MARIN, and in a simplified model that combines rudder and propeller into an abstract ‘thruster’. Steady states in the model correspond to circular motion of the ship and we compute the corresponding radii. We non-dimensionalize the models and thereby remove a number of parameters, so that, due to a scaling symmetry, only the rudder (or thruster) angle remains as a free parameter.

Using ‘degree theory’, we show that a slight modification of the model possesses at least one steady state for each angle and find certain constraints on the possible steady state configuration. We show that straight motion is unstable for the Hamburg test case and use numerical continuation and bifurcation software to compute a number of curves of states together with their stability, and the corresponding radii of the ship motion. In particular, straight forward motion can be stabilised by increasing the rudder size parameter, and the smallest possible radius is ~ 119 m.

These analyses illustrate methods and tools from dynamical systems theory that can be used to analyse a model without simulation. Compared with simulations, the numerical bifurcation analysis is much less time consuming. We have implemented the model in MATLAB and the bifurcation software AUTO.

1 Introduction

Traditionally, the study of the hydromechanic behaviour of a ship is divided into ship hydrostatics (without motions in calm water), and ship hydrodynamics (with motions in either calm water or in waves or current). The area of ship hydrodynamics can be roughly divided into powering/propulsion and calm water resistance, seakeeping (motions in waves with limited viscous effect) and manoeuvring (motion in calm water).

^{*}WUR, Wageningen

[†]TUe, Eindhoven

[‡]VU University Amsterdam

[§]Hogeschool Utrecht

[¶]Centrum voor Wiskunde en Informatica, Amsterdam

^{||}University of Leiden

^{**}KU Leuven

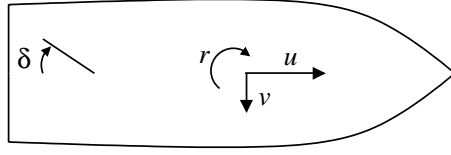


Figure 1: Ship-fixed coordinate system with respect to center of mass, and rudder angle notation.

Maneuvering research address the performances of a ship in typical operations such as a zig-zag manoeuvres, turning circles and harbor manoeuvres such as moving sideways or turning on the spot. The actual manoeuvring behaviour of a ship design is investigated in experiments using a scale model. The measured forces and moments are translated into coefficient values for the equations of motion. This allows for numerical simulations which can be used to investigate the manoeuvring behaviour for a variety conditions, such as speed and rudder angle.

In this report we rely on the model provided by MARIN in [7], which we refer to as the ‘Rudder model’ in the following. It accounts for the forces and moments that act on the center of mass of the ship by water, propeller and rudder. The model uses the reference system of the ship with velocities being surge u , sway v and yaw r . See Figure 1. All forces and moments are given with respect to center of mass, with the longitudinal force denoted by X directed forward according to u , the transverse force Y is directed to starboard according to v , and the rotational force N according to r .

1.1 Equations of motion

Due to the above setup, the framework model reads

$$\begin{aligned} (m + m_{uu})\dot{u} &= m r v + X_H + X_R + X_P \\ (m + m_{vv})\dot{v} + m_{vr}\dot{r} &= -m r u + Y_H + Y_R \\ m_{rv}\dot{v} + (I_z + m_{rr})\dot{r} &= N_H + N_R, \end{aligned} \quad (1)$$

where all quantities denoted by an m (and I_z) are ship dependent masses and moments of inertia, and the forces and moment X , Y , N have been decomposed into contributions from the hull (H), rudder (R) and propeller (P).

The equations of the force terms for the Rudder model from [7] are given next. These depend on a number of additional ship dependent quantities, such as ρ , L_{pp} , $X'_{u|u}$, \dots , for which we refer to [7]. In terms of u , v , r the forces and moment read

$$X_H = \frac{1}{2} \rho L_{pp} T \left(X'_{u'|u'} |u| |u| + X'_{\beta\gamma} L_{pp} v r \right) \quad (2)$$

$$Y_H = \frac{1}{2} \rho L_{pp} T \left(Y'_{\beta} |u| v + Y'_{\gamma} L_{pp} u r + Y'_{\beta|\beta} |v| |v| + Y'_{\gamma|\gamma} L_{pp}^2 r |r| \right. \\ \left. + Y'_{\beta|\gamma} L_{pp} v |r| + Y'_{|\beta|\gamma} L_{pp} |v| r + Y'_{ab} |u^{ay} v^{by}| \text{sign}(v) V^{-ay-by+2} \right) \quad (3)$$

$$N_H = \frac{1}{2} \rho L_{pp}^2 T \left(N'_{\beta} u v + N'_{\gamma} L_{pp} r |u| + N'_{u'\gamma c} L_{pp}^{c_n} |u r^{c_n}| V^{-c_n+1} \text{sign } r \right. \\ \left. + N'_{\gamma|\gamma} r |r| L_{pp}^2 + N'_{\beta|\beta} |v| |v| + N'_{\beta\beta\gamma} r v^2 L_{pp} V^{-1} \right. \\ \left. + N'_{\beta\gamma\gamma} v r^2 L_{pp}^2 V^{-1} \text{sign } u + N'_{ab} |u^{a_n} v^{b_n}| V^{-a_n-b_n+2} \text{sign}(u v) \right) \quad (4)$$

$$X_P = (1-t)T_p(u), \quad T_p(u) = \sum_{i=0}^5 K_{T_i} \left(\frac{u(1-w)}{nD_p} \right)^i \rho n^2 D_p^4. \quad (5)$$

$$X_R = -\frac{1}{2} \rho V_{rr}^{-1} A_R C_L \left(\frac{C_L u_r}{\pi \Lambda} (u_r \sin \delta - v_r \cos \delta)^2 \right. \\ \left. + v_r (u_r \sin \delta - v_r \cos \delta) (u_r \cos \delta + v_r \sin \delta) \right) \quad (6)$$

$$Y_R = \frac{1}{2} (1 + a_H) \rho V_{rr}^{-1} A_R C_L \left(u_r (u_r \sin \delta - v_r \cos \delta) (u_r \cos \delta + v_r \sin \delta) \right. \\ \left. - \frac{C_L v_r}{\pi \Lambda} (u_r \sin \delta - v_r \cos \delta)^2 \right) \quad (7)$$

$$N_R = Y_R x_r - X_R y_r \quad (8)$$

where

$$V_{rr} = \sqrt{u_r^2 + v_r^2} \\ u_r = u_p + C_{rue} \left(\sqrt{u_p^2 + \frac{8T_p(u)}{\rho \pi D_p^2}} - u_p \right) \\ v_r = C_{db} v + C_{dr} x_r r \\ u_p = (1-w)u$$

Strictly speaking, the rudder forces are given for the case of forward speed with positive thrust. For other courses these need to be modified.

Note that while u, v are velocities with dimension m/s, the third component r is an angular velocity with dimension 1/s. The first two force equations (1)₁ and (1)₂ are therefore nondimensionalised by the factor $\frac{1}{2} \rho V^2 L_{pp} T$ with $[\rho] = \text{kg/m}^3$, $[V] = \text{m/s}$, $[L_{pp}] = [T] = \text{m}$. The moment equation (1)₃ is therefore nondimensionalised by the factor $\frac{1}{2} \rho V^2 L_{pp}^2 T$, since $[N_i] = \text{N} \cdot \text{m}$. This yields a system of non-dimensional quantities of the form

$$\mathbf{M} \dot{\mathbf{w}} = \mathbf{f}(\mathbf{w}). \quad (9)$$

Remark. For the analysis of the equations of motion it is important to note that the model becomes invalid near $u = 0$. Indeed, the hull moment N_H is discontinuous due

to the term

$$L_{pp}^2 N'_{\beta\gamma\gamma} \text{sign}(u) v R^2 / V, \quad (10)$$

which is inconvenient for the analysis, and partly for the numerics as it causes degeneracies.

Moreover, according to the model given in [7], the form of the rudder forces actually changes for backward motion. Hence, solutions to the model used in this report with negative or vanishing u are not necessarily meaningful for the practical application. However, we mostly ignore this aspect as we strive for a theoretical analysis and the demonstration of our methods in this report.

1.1.1 Thruster model

Due to the complexity of the full ‘Rudder model’ and in order to isolate the influence of the hull forces, we introduce a simplified “Thruster model,” where the propeller and rudder forces are combined into an effective force acting on the hull that may be interpreted as a thruster.

Thus we replace $X'_P + X'_R$ and Y'_R, N'_R by abstract forces X'_T, Y'_T, N'_T , respectively, where

$$\begin{aligned} X'_T &= \tau \cos \alpha \\ Y'_T &= \tau \sin \alpha \\ N'_T &= x'_r \tau \sin \alpha, \end{aligned}$$

which means there is force of amplitude τ acting at angle α on the hull. The (non-dimensionalized) equations of motion read

$$\begin{aligned} (m' + m'_{uu})\dot{u} &= m' R v + X'_H + \tau \cos(\alpha) \\ (m' + m'_{vv})\dot{v} + m'_{vr}\dot{R} &= -m' R u + Y'_H + \tau \sin(\alpha) \\ m'_{rv}\dot{v} + (I'_z + m'_{rr})\dot{R} &= N'_H + x'_r \tau \sin(\alpha). \end{aligned} \quad (11)$$

where $R = L_{pp}r$. A notable difference to the full Rudder model is that here the propeller acts in a direction given by α , while the propeller in the Rudder model always pushes at angle zero. Moreover, the Rudder model does not distinguish rudder angles that differ by 180° ; the forces are the same, whether rudder points towards the stern or the aft. This is not so in the Thruster model, and indeed the results are quite different for angles around 180° .

2 Theoretical analysis

2.1 Interpretation of equations and steady states

Written compactly, the model equations are of the form

$$\mathbf{M}\dot{\mathbf{w}} = \mathbf{f}(\mathbf{w}), \quad (12)$$

which is a so-called algebro-differential equation as the left hand side is multiplied by a matrix. We will assume that, as in the Hamburg test case, the mass matrix is invertible so that we can rewrite (12) as the ordinary differential equation system

$$\dot{\mathbf{w}} = \mathbf{M}^{-1}\mathbf{f}(\mathbf{w}). \quad (13)$$

(If the mass matrix is not invertible the analysis become much more subtle.)

Steady states (also referred to as *equilibria*) of the differential equations are such that the time derivatives vanishing, that is, equilibria solve the non-linear algebraic equation

$$0 = \mathbf{f}(\mathbf{w}). \quad (14)$$

The set of solutions to this equation can be rather complicated and cannot easily be determined. Indeed, much of this report is dedicated to the existence and structure of solutions to (14).

A special steady state solution can, however, be found immediately: Since the equations model ship motion, for a straight rudder $\delta = 0$ or ($\alpha = 0$ for the Thruster model), there is a steady state where sway and yaw are zero ($r = v = 0$) –yielding straight motion– and the surge is adjusted according to propeller or thruster settings. See Section 2.2.2 for details on this course for the Thruster model.

The question arises what the course of the ship is at other steady states. This requires to change from the ship reference system to a reference system (x, y) of a standing observer. It turns out that, for a ship in a steady state (14), the observer can always be placed so that a ship that moves in a circle around the observer. To see this intuitively, note that the sum of the constant surge and sway generates drift in a fixed direction. The yaw generates a superimposed rotation so that the overall motion is a pure rotation.

Mathematically, it is convenient to use complex numbers and write $w = u + iv$ for the drift term and $z = x + iy$ for the observer coordinates. Let ϕ denote the orientation angle of the ship. Then

$$\dot{z} = we^{i\phi} \quad (15)$$

$$\dot{\phi} = \omega := r + \arg(w), \quad (16)$$

and so $\phi = \omega t + \phi(0)$, which gives

$$z = z(0) + \left(\frac{we^{i\phi(0)}}{i\omega} \right) e^{i\omega t}, \quad (17)$$

which is circular motion with radius $|w|/\omega$ and angular velocity ω .

Having found a steady state, the question is whether the ship can by itself follow this course, that is, whether perturbations from the steady state decay or at least do not grow. This is referred to as *stability* of the steady state. For ordinary differential equations stability of equilibria is essentially determined by the eigenvalues of the linearization of the right hand side evaluated in the steady state. In (13) this linearisation is the Jacobian matrix $\mathbf{M}^{-1}Df$. It is well known in nonlinear dynamical systems theory that:

- If the real parts of all eigenvalues are negative, then the equilibrium is stable: all small perturbations decay exponentially in time (with rate given by the largest of the negative real parts).

- If one of the eigenvalues has positive real part, then the steady state is unstable in the sense that all generic perturbations will drive the solution away from the equilibrium.
- The marginal case when all real parts are non-positive with one or more on the imaginary axis typically gives rise to a bifurcation. Roughly speaking bifurcation means that the set of bounded solutions changes qualitatively.

Rudder model. Before focussing on the simpler Thruster model, we remark that, for small propeller and rudder forces, the results on existence and stability of straight motion from the Thruster model carry over to the Rudder model. For instance, if the rudder area and propeller are small compared with $L_{pp}T$, then the hull forces dominate. For instance, in the numerical analysis of the Rudder model we find that straight motion is unstable, which we can show by pencil and paper for the Thruster model.

We briefly consider the ‘turn on the spot’ maneuver, which means $u = v = 0$, $R \neq 0$. As we will see in the next section, this maneuver is typically impossible in equilibrium for the Thruster model. Here we show that the same holds for the Rudder model with rudder at the symmetry axis of the hull, that is, $y'_r = 0$.

For $u = v = 0$, the first equation, (1) implies that $X'_P + X'_R = 0$. The second and third equation imply $0 = Y'_R + Y'_{\gamma|\gamma|}R|R|$, $0 = N'_R + N'_{\gamma|\gamma|}R|R|$. The symmetric rudder location $y'_r = 0$ implies $N'_R = Y'_R x'_r$ and we infer

$$Y'_R(Y'_{\gamma|\gamma|}x'_r - N'_{\gamma|\gamma|}) = 0.$$

Now, $Y'_R = 0$ implies $R = 0$, which corresponds to a static equilibrium. On the other hand, $Y'_{\gamma|\gamma|}x'_r \neq N'_{\gamma|\gamma|}$ for typical values of these ship dependent constants, which means that this maneuver is typically impossible in equilibrium.

2.2 Thruster model

In this section we analyze some aspects of the Thruster model. On the one hand we discuss stability and bifurcation of the simple straight forward motion, show non-existence of ‘turn on the spot’, and in particular discuss an abstract way to obtain insight into existence of equilibria. Due to the abstract nature of the model, we are not concerned with units and comparison with realistic values for the solutions we find.

We start out with noting symmetries of the Thruster model equations. The (u, r, v) -dependent terms on the right hand side are all homogeneous of degree 2 in the sense that rescaling $(u, R, v) \rightarrow \lambda(u, R, v)$ with $\lambda \geq 0$ yields the same terms multiplied by λ^2 . This means that for $\lambda = \sqrt{\tau}$, a time rescaling removes the parameter τ from the problem – except when it is zero, which we exclude in the following.

In addition to this scaling symmetry, the equations possess the symmetry

$$(\alpha, u, v, r) \rightarrow (-\alpha, u, -v, -r), \quad (18)$$

which means that any equilibrium has a symmetric partner for opposite thruster angle and is a result of the effective reflection symmetry of hull and thruster.

2.2.1 Non-existence of “turn on the spot”

In this section we show that as in the Rudder model, equilibria with $u = v = 0$ and $R \neq 0$ typically do not exist. Again we set $\dot{u} = \dot{v} = \dot{R} = 0$ in (11) to identify equilibria.

Substituting $u = v = 0$ into the first equation of (11) gives $0 = \tau \cos(\alpha)$, which implies either $\tau = 0$ (which is uninteresting) or $\alpha = (2k + 1)\pi/2$ for some integer k (which means a $\pm 90^\circ$ rudder angle).

Substituting $u = v = 0$ in the second and third equation implies

$$\begin{aligned} 0 &= Y'_{\gamma|\gamma|} R|R| + \tau \sin(\alpha) \\ 0 &= N'_{\gamma|\gamma|} R|R| + x'_r \tau \sin(\alpha). \end{aligned}$$

The case $\tau = 0$ immediately implies $R = 0$, and for the case $\alpha = (2k + 1)\pi/2$, we have $R|R| = -\tau \sin(\alpha)/Y'_{\gamma|\gamma|}$, which yields the condition

$$N'_{\gamma|\gamma|}/Y'_{\gamma|\gamma|} = x'_r,$$

which already appeared in the Rudder model. For generic values of the ship constants this is not true. Theoretically, one may view x'_r as an independent parameter so that the above constraint shows where to put the rudder in order to have a ship that is able to perform pure rotational motion in equilibrium for fully sideways thruster.

2.2.2 Stability of straight motion

Straight motion in equilibrium is a solution to (11) with vanishing left hand side and $v = r = 0$. Equations (11)_{2,3} imply $\alpha = 0$ and from (11)₁ it follows that $u = u(\tau)$ is the unique solution to

$$X'_{u'|u'}|u| + \tau = 0.$$

Hence, for $X'_{u'|u'} \neq 0$, $u = 0$ if and only if $\tau = 0$, which we do not consider here. In particular,

$$\text{sign}(u) = -\text{sign}(\tau X'_{u'|u'}), \quad |u| = \sqrt{|\tau/X'_{u'|u'}|}.$$

The linearisation in $r = v = 0$ of the right hand side of (11) reads

$$\begin{pmatrix} \partial_u X'_H & 0 & 0 \\ 0 & \partial_v Y'_H & \partial_r Y'_H \\ 0 & \partial_v N'_H & \partial_r N'_H \end{pmatrix} = \begin{pmatrix} 2X'_{u'|u'}|u| & 0 & 0 \\ 0 & Y'_\beta|u| & (Y'_\gamma - m')u \\ 0 & N'_\beta u & N'_\gamma|u| \end{pmatrix}$$

where partial derivatives are evaluated at $r = v = 0$.

In order to derive stability properties this must be multiplied by the inverse of the mass matrix on the left hand side of (11) given by

$$\begin{pmatrix} m' + m'_{uu} & 0 & 0 \\ 0 & m' + m'_{vv} & m'_{vr} \\ 0 & m'_{rv} & I'_z + m'_{rr} \end{pmatrix}^{-1} = D^{-1} \begin{pmatrix} D(m' + m'_{uu})^{-1} & 0 & 0 \\ 0 & I'_z + m'_{rr} & -m'_{rv} \\ 0 & -m'_{vr} & m' + m'_{vv} \end{pmatrix}.$$

Here $D = (m' + m'_{vv})(I'_z + m'_{rr}) - m'_{rv}m'_{vr}$ is the determinant of the lower right 2-by-2 submatrix of the mass matrix for (r, v) . For the Hamburg test case, $D = 0.06$. In that case also $X'_{u'|u'} < 0$, which is natural as it means that the acceleration in the hull direction will adjust to the force balance given by the thrust that acts in that direction; at least on the linear level.

Therefore, stability is determined by the (r, v) -submatrix, which measures the effects of perturbations in the v - and r -directions. The matrix reads

$$S := \begin{pmatrix} I'_z + m'_{rr} & -m'_{rv} \\ -m'_{vr} & m' + m'_{vv} \end{pmatrix} \begin{pmatrix} Y'_\beta |u| & (Y'_\gamma - m')u \\ N'_\beta u & N'_\gamma |u| \end{pmatrix} = \\ \begin{pmatrix} (I'_z + m'_{rr})Y'_\beta |u| - m'_{rv}N'_\beta u & (I'_z + m'_{rr})(Y'_\gamma u - m') - m'_{rv}N'_\gamma |u| \\ -m'_{vr}Y'_\beta |u| + (m' + m'_{vv})N'_\beta u & -m'_{vr}(Y'_\gamma - m')u + (m' + m'_{vv})N'_\gamma |u| \end{pmatrix}.$$

Trace and determinant are

$$\begin{aligned} \text{tr}(S) &= ((I'_z + m'_{rr})Y'_\beta + (m' + m'_{vv})N'_\gamma)|u| - (m'_{rv}N'_\beta + m'_{vr}(Y'_\gamma - m'))u \\ \det(S) &= D(Y'_\beta N'_\gamma - N'_\beta(Y'_\gamma - m'))u^2. \end{aligned}$$

The masses are always positive. For the Hamburg test case these and the hydrodynamic hull constants are

$$\begin{aligned} m' &= 0.2328, m'_{vv} = 0.2286, m'_{rv} = m'_{vr} = 0.0074, I'_z + m'_{rr} = 0.0284, \\ Y'_\beta &= -0.1735, Y'_\gamma = 0.0338, N'_\beta = -0.1442, N'_\gamma = -0.0267. \end{aligned}$$

In particular, (keeping four digits)

$$\begin{aligned} \text{tr}(S) &= (-\text{sign}(u)0.0174 + 0.0025)u < 0, \\ \det(S) &= 0.0601(-0.0240)u^2 = -0.0014u^2 < 0. \end{aligned}$$

Since the determinant (product of eigenvalues) is negative, eigenvalues are real with opposite signs. The negative trace means that the positive eigenvalue is smaller than the negative in absolute values. In this case eigenvalues are (to three digits) -0.027 and 0.012 .

In particular, the positive eigenvalue means that equilibrium straight motion for the Hamburg test case is *unstable*, which makes sense for a hull to aid manoeuvring. The eigenvector associated to the unstable eigenvalue 0.012 is $(0.331, -0.94)$, which means that the main effect of the instability is a rotation in negative r direction; the smaller effect is a sideways shift in positive v -direction.

2.2.3 Theoretical stability change

At least from a theoretical viewpoint, it is instructive to study which of the ship parameters can change stability of straight motion. It turns out that one way is by varying m' only, keeping other values the same. In practice this may be impossible as other ship constants may change when m' changes. Nevertheless, this sheds light on the possible bifurcation structures.

Trace and determinant in that case read

$$\begin{aligned} \text{tr}(S) &= (-\text{sign}(u)(0.0111 + 0.0267m') + 0.0008 + 0.0074m')u < 0, \\ \det(S) &= (0.00003 + 0.0017m' - 0.0350(m')^2)u^2. \end{aligned}$$

Since the trace remains always negative, a stability change must occur at vanishing determinant. Recall that positive determinant for negative trace means both eigenvalues are negative (or have negative real part) and therefore imply stability of the equilibrium straight motion.

Since the determinant is quadratic in m' and positive at $m' = 0$, it is positive only in an interval including $m' = 0$, and negative for all other values of m' . The Hamburg test case has negative determinant and therefore m' must be decreased to generate a stable equilibrium straight motion. This may be counter-intuitive as this suggest decreasing hulls weight generates stability. However, in practice it may be impossible to change m' alone as modification of the design likely effect other parameters as well.

In nonlinear systems such as (11), a stability change by a real eigenvalue implies bifurcation of other equilibrium solutions as discussed in the next section.

2.3 Stability analysis of the simple thruster model

The equation of motion are

$$\begin{aligned}(m + m_{uu})u' &= mrv + X_G \\ (m + m_{vv})v' + m_{vr}r' &= -mru + Y_G \\ m_{rv}v' + (I_z + m_{rr})r' &= N_G,\end{aligned}$$

where $m_{vr} = m_{rv}$. Here $(X_G, Y_G, N_G) = (X_H + X_T, Y_H + Y_T, N_H + N_T)$ are composed of the hull forces/moments and thruster forces/moments. The former are given, in slightly simplified form, by

$$\begin{aligned}X_H &= X_{uu}u|u| + X_{\beta\gamma}vr \\ Y_H &= Y_{\beta}|u|v + Y_{\gamma}ur + Y_{\beta\beta}v|v| + Y_{\gamma\gamma}r|r| + Y_{\beta\gamma}v|r| + Y_{\gamma\beta}|v|r \\ N_H &= N_{\beta}uv + N_{\gamma}r|u| + N_{\gamma\gamma}r|r| + N_{\beta\beta}v|v|.\end{aligned}$$

In our simple thruster model we have

$$\begin{aligned}X_T &= \tau \cos \alpha \\ Y_T &= \tau \sin \alpha \\ N_T &= \tau x_{\tau} \sin \alpha,\end{aligned}$$

where x_{τ} is the position of the thruster (relative to the center of mass), τ is the magnitude of the trust force, and α its angle.

For $\alpha = 0$ we linearize around the straight motion $(u, v, r) = (u_0, 0, 0)$, with $u_0 = \sqrt{-\tau/X_{uu}}$, where $X_{uu} < 0$. The linearized equations are

$$\begin{bmatrix} m + m_{uu} & 0 & 0 \\ 0 & m + m_{vv} & m_{vr} \\ 0 & m_{rv} & I_{zz} + m_{rr} \end{bmatrix} \begin{bmatrix} u' \\ v' \\ r' \end{bmatrix} = \begin{bmatrix} 2X_{uu}u_0 & 0 & 0 \\ 0 & Y_{\beta}u_0 & (-m + Y_{\gamma})u_0 \\ 0 & N_{\beta}u_0 & N_{\gamma}u_0 \end{bmatrix} \begin{bmatrix} u \\ v \\ r \end{bmatrix}.$$

Clearly, the stability of $(u_0, 0, 0)$ depends on the matrix

$$\begin{bmatrix} Y_\beta & (-m + Y_\gamma) \\ N_\beta & N_\gamma \end{bmatrix}. \quad (19)$$

Typical (suitably nondimensionalized) values for the parameters are

$$Y_\beta = -0.2, \quad Y_\gamma = 0.03, \quad N_\beta = -0.1, \quad N_\gamma = -0.03, \quad m = 0.2,$$

for which the determinant and trace of (19) are negative: this means that this 2×2 submatrix has one positive and one negative eigenvalue. Thus, the solution $(u, v, r) = (u_0, 0, 0)$ is unstable. For smaller values of m the determinant becomes positive, while the trace remains negative. Hence, the point $(u_0, 0, 0)$ can become stable if one changes parameters.

We remark that the ship's left-right symmetry implies that its equations of motion are symmetric under the map $S : (u, v, r) \mapsto (u, -v, -r)$, see (18).

The generic scenario in which a stationary point of a smooth vector field with the above symmetry loses stability is the pitchfork bifurcation. Because the equations of motion of the ship are not smooth - due to the dissipative terms of the form $u|u|$ and so on - a model for the bifurcation in question is described by the local normal form

$$\frac{dx}{dt} = f_\pm(x, \lambda), \text{ with } f_\pm : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R} \text{ defined by } f_\pm(x, \lambda) := x(\lambda \pm |x|).$$

Note that the symmetry of this differential equation is reflected by $f_\pm(-x, \lambda) = -f_\pm(x, \lambda)$.

It is clear that $x = 0$ is an equilibrium point for f_\pm for all values of the parameter λ . The linearized differential equation $\frac{dx}{dt} = \lambda x$ shows that this point is stable for $\lambda < 0$ and unstable for $\lambda > 0$. Moreover, on one side of the bifurcation point $\lambda = 0$, two extra stationary solutions exist, given by $x = \pm\lambda$. These branches emerge from the point $(x, \lambda) = (0, 0)$. For f_- they exist for $\lambda > 0$, which is why f_- is called a “supercritical pitchfork”. These additional equilibria are stable. f_+ is called a subcritical pitchfork, because the extra solutions exist for $\lambda < 0$. They are unstable. Figure 2 is the bifurcation diagram for f_- , that summarizes this information. Figure 2 should be compared with numerical results that were obtained for the thruster model. It shows a nearly exact match with the theory.

For completeness, we have also indicated the local degree of the solutions. Note that the total degree is always equal to -1 . The concept of degree will be explained in the next section. It will also be explained in the next section that an unstable straight motion can never exist alone. This provides a topological explanation of why in the above non-smooth pitchfork bifurcation, two stable solutions are born.

2.4 Degree theory: Existence of Equilibria

The ship models we have considered so far can be written in the following general form

$$M\dot{\vec{v}} = F(\vec{v}, \alpha) = F^\perp(\vec{v}) + F_D(\vec{v}, \alpha) + F_T(\vec{v}, \alpha),$$

where $\vec{v} = (u, v, Lr)$, M is an invertible symmetric matrix with the dimension of mass, $F^\perp(\vec{v}) = (mrv, -mru, 0)$ is the Coriolis force coming from the choice of

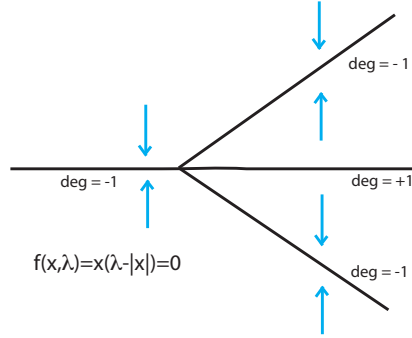


Figure 2: Bifurcation diagram of $\frac{dx}{dt} = f_-(x, \lambda) = x(\lambda - |x|)$.

rotating coordinates, F_D is a dissipative force from friction and viscosity, and F_T is a thrusting force. We have a certain freedom in the way we split up a force in a F_D and F_T part. In particular, for the rudder model there is a term $F_D(\vec{v}, \delta)$ and a thruster force $F_T(u)$ which is a bounded function of u , whereas in the steerable thruster model there is the assumption that $F_D = F_D(\vec{v})$ depends only on the velocity and $F_T = F_T(\alpha)$ is a constant that only depends on a steering angle. All forces implicitly depend on design parameters. We sometimes suppress the steering angle α , as for our purposes it can be considered a fixed parameter just like the design parameters.

There is a fundamental physical reason for splitting up the forces. We make assumption that for very large velocities the dissipative force will give a kinetic energy loss that dominates the thrust force. Of course in this statement the precise meaning of very large may depend on parameters like the power generated by the engines. Thus, this assumption is very mild. Since the mass matrix is symmetric, the change in kinetic energy per second can be computed as

$$\begin{aligned} \frac{d}{dt} E_{kin} &= \frac{1}{2} \frac{d}{dt} \vec{v} \cdot M \vec{v} \\ &= \frac{1}{2} (\dot{\vec{v}} \cdot M \vec{v} + \vec{v} \cdot M \dot{\vec{v}}) \\ &= \vec{v} \cdot M \dot{\vec{v}} \\ &= \vec{v} \cdot (F_D + F_T), \end{aligned}$$

where in the last line we used that the Coriolis force is orthogonal to the velocity vector \vec{v} .

A precise formulation of the assumption we make is the following. For $V_0 > 0$ let

$$S^2(V_0) = \{\vec{v} \in \mathbb{R}^3 \mid \|\vec{v}\| = V_0\}$$

be the 2-sphere with radius V_0 . It bounds the 3-ball $B^3(V_0) = \{\vec{v} \in \mathbb{R}^3 \mid \|\vec{v}\| \leq V_0\}$. Here it is easiest to think of the normal round sphere but if we wish, we can choose the norm $\|\cdot\|$ to define the sphere $S^2(V_0)$ at our convenience e.g. we could define $\|\vec{v}\| = \max(|u|, 12.34|v|, 5.678|Lr|)$ if the region of interest is defined by different upper bounds for forward, transverse and yaw velocities.

Assumption 2.1. *If V_0 is sufficiently large then for all velocities $\vec{v} \in S^2(V_0)$ the kinetic energy decreases i.e.*

$$\vec{v} \cdot F = \vec{v} \cdot (F_D + F_T)(\vec{v}) < 0$$

We should explain what sufficiently large means. The condition is physically very natural, if not inevitable. For the model, however, it suffices that the assumption is true for V_0 somewhat larger than the velocities of a ship at maximum thrust. We really only assume that the model is physically acceptable for high, but not for insanely high velocity ranges. For example, if the model contains small high order polynomial corrections that, when blindly extrapolated to absurdly high velocities, become leading order and ruin the assumption, that does not matter. In fact, it is best to assume it is only defined for $\|\vec{v}\| \leq V_0$. Mathematically the point is that on the boundary of the velocity range of interest the force field is pointing inwards. We also assume that it is topologically a three-dimensional ball bounded by a 2-sphere, but that could be relaxed.

2.4.1 Existence result

The flow $\Phi_F^t(\vec{v})$ of the vector field $F(\vec{v})$ is the solution of the equation $\dot{w} = F(w)/kg$ with boundary condition $w_0 = \vec{v}$. If the vector field is continuous its flow is also continuous for short times. The fixed points of the flow for $t > 0$ are exactly the zeros of F i.e. the equilibria. Now a reformulation of assumption 2.1 is that F is inward pointing on the sphere $S^2(V_0)$. It follows that the flow Φ_F^t can never leave the ball $B^3(V_0)$ and maps the ball to itself, i.e. for fixed t the flow can be restricted to a continuous map

$$\Phi_F^t : B^3(V_0) \rightarrow B^3(V_0).$$

By the Brouwer fixed point theorem, a continuous map from the ball to itself always has a fixed point. Therefore, an equilibrium always exists.

2.4.2 There are -1 Equilibria

The Brouwer fixed point theorem is one of the first and best known examples of the use of algebraic topology in analysis. Using a tiny bit of algebraic topology directly we can make a much more precise quantitative statement about the number of equilibria *counted with multiplicity*. In particular we will be able to state what we mean by multiplicity and it turns out that negative multiplicities have to be allowed.

The counting of equilibria is a direct consequence of the following statements. They can all be considerably generalized, but the current statements suffice for our purposes.

Proposition 2.2. *Let $\Omega \subset \mathbb{R}^n$ be a region which is topologically a closed n -ball B^n with a piecewise smooth boundary $\partial\Omega$ which is topologically an $n - 1$ sphere S^{n-1} . For every continuous nowhere vanishing vector field $F : \partial\Omega \rightarrow \mathbb{R}^n - \{0\}$ there is a well-defined degree $\deg(F) \in \mathbb{Z}$ with the following properties.*

1. The degree is constant in every continuous 1 parameter family F_s of non-vanishing vector fields i.e. for every continuous map¹

$$\hat{F} : \partial\Omega \times [0, 1] \rightarrow \mathbb{R}^n - \{0\},$$

where $F_s = \hat{F}|_{\partial\Omega \times \{s\}}$ and $F_0 = F$, the degree $\deg(F_s) = \deg(F)$ for all $0 \leq s \leq 1$.

2. Suppose that $\tilde{F} : \Omega \rightarrow \mathbb{R}^n - 0$ is a continuous vector field which is continuously differentiable and transverse on the interior Ω° extending F , i.e. $\tilde{F}|_{\partial\Omega} = F$, the extension \tilde{F} vanishes in isolated points z_1, \dots, z_m and its Jacobian matrix $D\tilde{F} : \Omega \rightarrow \mathbb{R}^{n \times n}$ is invertible at the zeroes z_i , then

$$\deg(F) = \sum_i \text{sign}(\det(D\tilde{F}(z_i))).$$

Note that a transverse smooth extension \tilde{F} as in 2.2.2 always exists. These two statements allow us to count the number of equilibria. We use the notation of the beginning of section 2.4.2. For a proof of proposition 2.2, we refer to Appendix A.

We can define the number of equilibria counted with multiplicity as

$$\deg(F|_{S^2(V_0)})$$

for a large enough velocity V_0 . See the discussion below assumption 2.1 for the meaning of large enough in the physical model. By proposition 2.2.2 we see that the degree has an interpretation as a number equilibria counted with multiplicity if the force vector field on the space of velocities is transverse, but we can always perturb F by an arbitrary small perturbation to get a transverse extension \tilde{F} . We also see that in the transverse case it is independent of V_0 , as long as changing V_0 does not introduce new zeroes. We can always assume that the zeros of the transverse perturbation \tilde{F} are all in a small neighborhood of the zeros of F to see that this is true in general.

We can compute the degree of $F|_{S^2(V_0)}$ using assumption 2.1 by deforming F to a vector field of which it is easier to compute the degree. We write v instead of \vec{v} . Let $F^{op} : v \rightarrow -1v$ be the force vector field which is always directed opposite to the velocity with a unit of proportionality that is 1 in suitable units. It is obviously nonzero on $S^2(V_0)$. Now define a one-parameter family $\hat{F} : S^2(V_0) \times [0, 1] \rightarrow \mathbb{R}^3 - 0$ deforming F into F^{op} by

$$\hat{F}(v, s) = (1 - s)F(v) + sF^{op}(v) \quad (20)$$

It is obviously continuous and well defined as a family of vector fields. The reason it is non-vanishing on $S^2(V_0)$ is that for all $v \in S^2(V_0)$

$$v \cdot F_s = (1 - s)v \cdot F(v) + sv \cdot F^{op}(v) = (1 - s)v \cdot F(v) - sv \cdot v < 0 \quad \forall s \in [0, 1]$$

We conclude that $\deg(F|_{S^2(V_0)}) = \deg(F^{op}|_{S^2(V_0)})$ and we only have to compute the latter. We can apply 2.2.2 to $F^{op}|_{S^2(V_0)}$ which is obviously already defined as a transverse vector field over the entire ball. Since we work in dimension 3

$$\deg(F^{op}|_{S^2(V_0)}) = \text{sign}(\det(-1)) = \text{sign}((-1)^3) = -1.$$

¹Topologists call such a map a homotopy between F_0 and F_1

Actually this is cheating a bit, because to prove formula 2.2.2 we will have to compute the degree of a linear vector field directly. In any case we conclude

Proposition 2.3. *Under the assumption 2.1 on the force vector field F , there are -1 equilibria counted with multiplicity on the ball $B^3(V_0)$ i.e.*

$$\deg(F|_{S^2(V_0)}) = -1.$$

3 Numerical analysis

3.1 Numerical Bifurcation Analysis of the Thruster Model

The scaling symmetry with respect to τ of the Thruster model means that the exact value of τ is not relevant. In addition, this is a toy model and so we somewhat randomly picked and fixed $\tau = 1000$.

All numerical computations in this section have been done with the continuation and bifurcation software AUTO [5]. We refer to the extensive manual for details. Here we only used the capabilities to solve algebraic equations with eigenvalue computation and detection of Hopf bifurcations, and to continue the branches of periodic solutions that emerge at such bifurcations. Fixing τ and the ship parameters from the ‘Hamburg test case’, the only parameter left in the Thruster model is the angle α , which is thus the primary bifurcation parameter in these computations.

Briefly, the idea of continuation is to numerically implement the implicit function theorem: given an equilibrium, parameter changes typically move the equilibrium along a smooth curve. In Section 3.3 we discuss a naive implementation of this idea with MATLAB. However, this naive approach fails at certain bifurcations and cannot resolve the bifurcating solutions, which is possible with AUTO. In particular, AUTO automatically computes the eigenvalues of the linearisation in the equilibria, which decide upon stability of the equilibrium. Hence, eigenvalue plots as given in Section 3.3 can also be made with AUTO, but we omit this here. In all plots of this section, solid lines correspond to stable solutions, dashed to unstable ones.

Notably, these steady state computations do not require to numerically solve the differential equation: we do not compute trajectories. The periodic solutions are computed automatically by the software. Therefore, these computations are extremely fast. For the simple three-dimensional ordinary differential equation used here the speed-up is modest, but it is dramatic for larger systems or partial differential equations. Moreover, this approach allows to detect unstable solutions, which is not possible by simulation. Knowing the location of unstable equilibria can be relevant, for instance, in order to find further stable equilibria or to apply a stabilizing control.

3.1.1 Stationary states by continuation from straight motion

In this section we present the result of continuation from the trivial straight motion equilibrium which was discussed in section 2.2.2. Recall symmetry (18) of (11), which will be inherited by the equilibrium branches – with branch we mean a curve of equilibria (or other types of solutions) and associated parameter values in the product of phase space and parameter (usually only α).

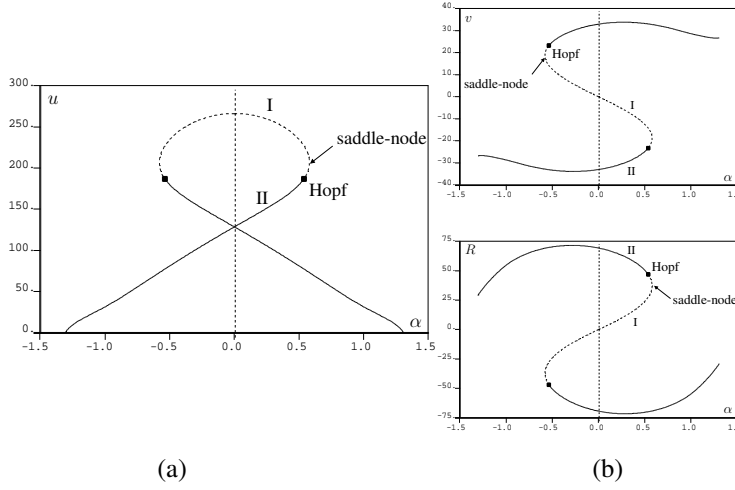


Figure 3: (a) The u -component of the branch of equilibrium solutions that connect to the straight motion. Dashed = unstable, solid = stable. Units of α are radians, units of u are m/s , but not meaningful for the random choice of $\tau = 1000$. (b) The other components of the equilibria shown in (a): upper panel v -components, lower panel R -components.

In Figure 3 we plot the numerical result from continuing the unstable straight motion equilibrium; this lies at the top of the dashed curve at $\alpha = 0$. In the following we describe the meaning of this figure and its interpretation for the ship motion, and refer to the labels I, II for the different sections along the branches in Figure 3(a).

Branch part I: Starting from the straight motion in either direction from $\alpha = 0$, the branch folds at $\alpha \approx \pm 0.58$, where saddle-node bifurcations take place and the equilibrium becomes unstable with respect to an additional real eigenvalues. The equilibrium is thus unstable in any (v, r) -direction. Further along the branch, these eigenvalues form a complex conjugate pair whose real parts cross zero at a Hopf-bifurcation.

Branch part II: Beyond the Hopf-bifurcation, the equilibria are stable and extend up to $\alpha \approx \pm 1.306$, which is about 75° . Notably, for each direction in α the branch crosses $\alpha = 0$ and thus there exist two stable equilibria at $\alpha = 0$, which do not have $v = r = 0$. Note that there are two stable and one unstable equilibria at $\alpha = 0$, which agrees with the predictions by degree theory 2.4.

At $u = 0$ the vector field is discontinuous, and it is therefore not surprising that the numerical continuation terminates at this value. Nevertheless, there exist other solution branches. See Section 3.1.3.

From Figure 3(b) we see that these equilibria have positive v - and negative r -components or vice versa. In particular, there are stable equilibria with positive thruster angle and rotation, which is at first counter-intuitive as the thrust points in the negative direction so that the sign should be opposite. Since the sideways velocity has this opposite sign, the hull motion is as expected, but the interaction of forces generates a rotation in the ‘wrong’ direction.

Thinking experimentally, this phenomenon occurs when first moving on an equilibrium circle with negative rudder angle and positive R on branch II, and then slowly increasing the rudder angle. The model predicts that even beyond straight rudder angle, the ship will still rotate in the direction it did before. However, when continuing this slow increase in α , at the Hopf-bifurcation the ship loses the ability to move in equilibrium and to rotate in the ‘wrong’ direction. In the next section we show that this is a subcritical bifurcation, which means that something ‘dramatic’ happens that the local bifurcation analysis cannot reveal. One possibility is that the motion settles to the other stable equilibrium with negative R and the overall dynamics follows a ‘hysteresis’ (see explanation below).

3.1.2 Periodic solutions emerging from primary Hopf bifurcation

In this subsection we discuss the branch of periodic solutions that bifurcates from the branch of steady states at the Hopf-bifurcation marked in Figure 3. See Figure 4. As mentioned, the bifurcation is subcritical, which means that the bifurcating periodic solutions are unstable so that the ship would not follow this trajectory by itself.

The branch of periodic solutions terminates at a homoclinic bifurcation, which means that the periodic profile ‘collides’ with an equilibrium: a homoclinic orbit is such that the trajectory converges in forward and backward time to the same equilibrium and makes an intermediate excursion. When approaching such a solution, the periodic orbits spend more and more time near equilibria which generates the characteristic profiles plotted in Figure 4(b). In the lower panel of this figure the gradients become large when the excursion from the equilibrium localises to a point because the period of all these plots is normalised to 1.

3.1.3 Other solution branches

The degree theory argument from Section 2.4 suggests that there exists at least one equilibrium for all angles α . Strictly speaking, this does not apply due to the discontinuity of the vector field at $u = 0$ and indeed, some new solutions are created when smoothing out the discontinuity as discussed at the end of this section. Nevertheless, the degree argument motivates to search for equilibria at angles $\alpha > 1.306$, which was the boundary from the previous section: $u = 0$ at $\alpha \approx 1.306$.

Using a root finder for larger values of α produces new solutions and thus the additional solution branches plotted in Figure 5. These are not connected to the branches already plotted in Figure 3 (due to the discontinuity at $u = 0$; see end of this section). Almost all of these solutions have negative u and may thus be less interesting for the application. One of the branches with positive u comes relatively close to the stable branch (marked by a circle) also in the r and v components, which suggests that the stable equilibria are somewhat more sensitive to perturbations for angles near $\alpha = 1.306$.

The rectangular region marked in Figure 5 contains several stable branches (with $u < 0$) and is enlarged in Figure 6(a). Notably, there co-exist two stable equilibria for all α between the vertical dotted lines, and these are connected by a branch of unstable equilibria. Such a configuration with two competing stable states is a signature of nonlinear systems, and also suggest vicinity of ‘cusp’ bifurcation. In practice this

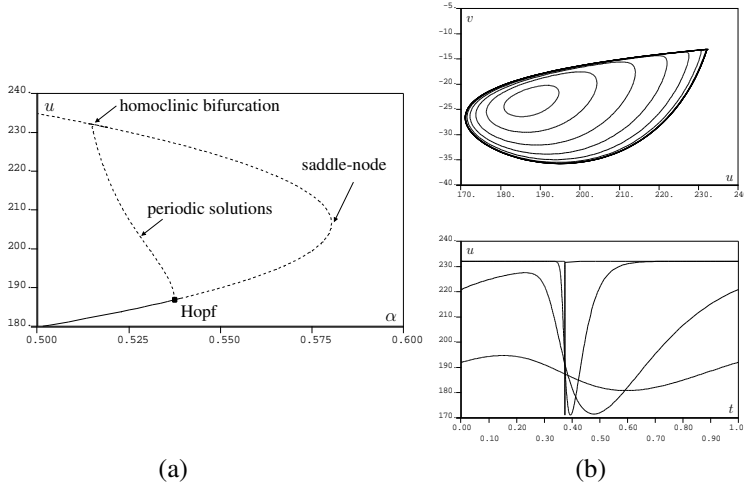


Figure 4: (a) Branch of periodic solutions that bifurcates from the steady state at the Hopf bifurcation, and terminates in a homoclinic bifurcation with background state at the upper end of the branch. (b) Plots of some periodic solutions along the branch in (a): upper panel (u, v) -space, lower panel (x, u) -space with period normalised to 1.

suggests that when starting with a ship making a stable circular motion from the upper branch and slowly increasing the thruster angle, the ship would suddenly jump (at the right dotted line) to another circular motion on the lower branch and the previous circular motion cannot be recovered by small modifications of the thruster angle. In order to jump back to the upper branch, the thruster angle would have to be moved back beyond the left dotted line. This phenomenon is called ‘hysteresis’.

The rectangular region marked in Figure 6(a) is enlarged in Figure 6(b). It shows a supercritical Hopf bifurcation, where a branch of stable periodic orbits bifurcates. This branch terminates in a homoclinic bifurcation analogous to the case discussed in Section 3.1.2. (The plot is discontinuous because it shows the maximum u -value of the periodic solutions, while the equilibrium of the homoclinic bifurcation lies at the minimum.)

The role of the discontinuity of the hull force can be illustrated by replacing $\text{sign}(u)$ with, e.g., $2\arctan(u/\varepsilon)/\pi$ for small $\varepsilon > 0$. In Figure 7 we plot a comparison of this smoothing for $\varepsilon = 0.01$ with the discontinuous case. The branches in Figure 5 that are disconnected for the discontinuous case are connected for the smoothed case, and it seems that there are no further solutions. This would mean that in the discontinuous case there are intervals in α , e.g., $\alpha \sim 1.75$, for which there do not exist solutions. On the other hand, the fact that non-trivial solutions with $u = 0$ occur at all may be an artefact of the Thruster model. For the Rudder model, the forward thrust of the propeller should make this impossible, and indeed we do not find such solutions, as discussed in the next section.

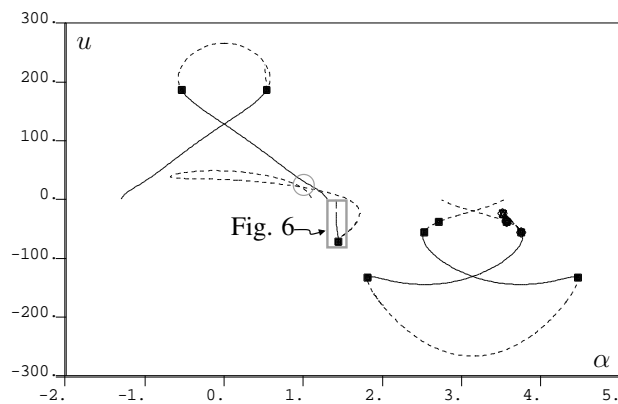
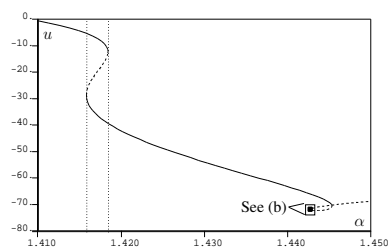
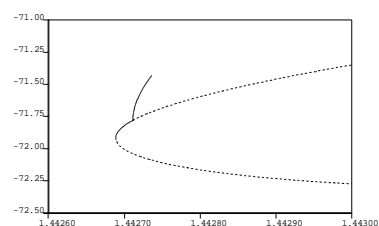


Figure 5: All the branches that have been found numerically. Additional branches arise from reflecting these about $\alpha = 0$.

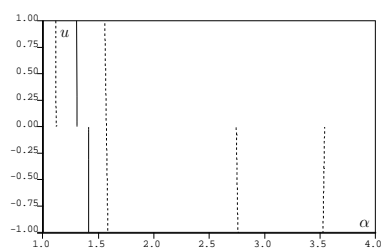


(a)

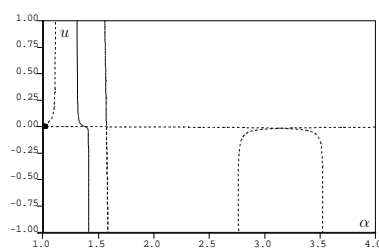


(b)

Figure 6: (a) Enlargement of the region marked in Figure 5. The S-shaped branch generates ‘bistability’: two different stable equilibria co-exist between the vertical dotted lines. (b) Enlargement of the region marked in (a). A branch of stable periodic orbits emerging in a Hopf bifurcation and terminating in a homoclinic bifurcation.



(a)



(b)

Figure 7: (a) Enlargement of a region near $u = 0$ of Figure 5. (b) The solutions for smoothed forces, replacing $\text{sign}(u)$ by $2\arctan(100u)/\pi$.

3.2 Numerical Bifurcation Analysis of the Rudder Model

Analogous to the computations of the previous section, here we use AUTO to analyse numerically the non-dimensional ‘Rudder model’. Note that in Section 3.3 below we present analogous, albeit less complete, results using a naive implementation of the continuation approach in MATLAB. The added value of the MATLAB routines is that these can automatically find equilibrium solutions (without a given good guess for one).

Similar to the scaling symmetry in τ of the Thruster model (11), here we have a scaling symmetry in the rate of rotation n of the propeller. Specifically, let $T_p(u; n)$ be the propeller force term T_p with explicit n -dependence. Then

$$T_p(\lambda u; \lambda n) = \lambda^2 T_p(u; n).$$

As mentioned for the Thruster model, the hull forces are homogenous of degree two, thus having the same scaling law as T_p . It is straightforward that also the rudder forces possess this homogeneity and therefore the entire right hand side does. Hence, it suffices to know solutions for one value of n , as the result for any other value follows from the above rescaling, as long as $n \neq 0$. For the numerical calculations we therefore fix $n = 2$.

In contrast to the Thruster model, this model has the two reflection symmetries

$$\begin{aligned} (\delta, u, v, R) &\rightarrow (-\delta, u, -v, -R), \\ (90^\circ + \delta, u, v, R) &\rightarrow (90^\circ - \delta, u, -v, -R). \end{aligned} \quad (21)$$

In particular, the model does not distinguish between the orientations of the rudder for a given angle. Due to the symmetries (21), the bifurcation diagrams are reflection symmetric about both $\delta = 0$ and $\delta = 90^\circ$ (and thus also about 180°). Another effect of neglecting the rudder orientation is that the branches of equilibria encountered here do not (need to) cross $u = 0$ so that these stay within the range of validity of the model. Recall N'_H is discontinuous at $u = 0$, and the branches in Figure 5 include $u = 0$. Indeed, the Thruster model (11) accounts for the direction of the rudder/thruster, and the branches in Figure 5 very roughly have the symmetry $(\alpha, u, r, v) \rightarrow (\pi + \alpha, -u, -r, -v)$.

3.2.1 Continuation from straight motion, and radii of rotation

In this section we use the parameter values of the Hamburg test case.

As for the Thruster model, a natural starting point for investigating equilibria is the straight motion where $\delta = v = r = 0$. In Figure 8 we plot the resulting branch of equilibria when varying δ . For small δ the result is qualitatively the same as for the Thruster model: straight motion is unstable and beyond folds (alias saddle-node bifurcations) around $\delta \approx \pm 1.5^\circ$ there coexists a symmetric pair of stable branches ‘bistability’. However, this range of angles is much smaller than in the Thruster model and there is no Hopf-bifurcation.

Branch part I: Beyond the folds, the branch is stable and has monotonically decreasing u -value until $\delta \approx \pm 75^\circ$, while r and v behave non-monotonically both having a single extremum at $\delta \approx 28.5^\circ$ (minimum for v , maximum for r). In contrast to the Thruster model there is no Hopf bifurcation in this range.

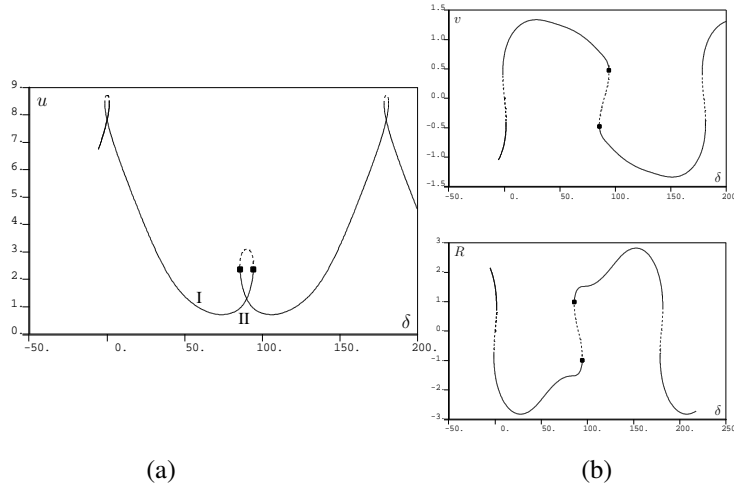


Figure 8: Branch of equilibria connected to the straight motion. (a) The u -component, (b) the v -component (upper panel) and R -component (lower panel). Recall that the components have units of m/s . The full branch consists of this part and the reflection about $\delta = 0$. The branch also has a reflection symmetry about $\delta = 90^\circ$. Solid lines denote stable equilibria, dashed unstable ones. Squares denote Hopf bifurcations.

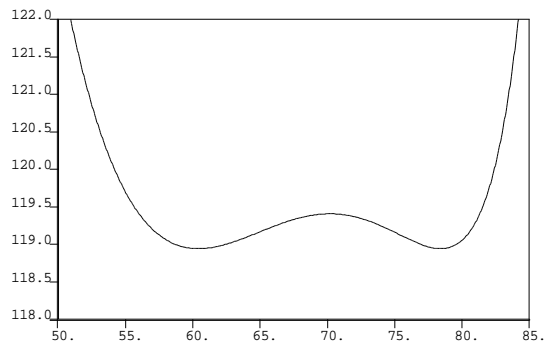


Figure 9: Radii, in m, of circle motion corresponding to part of the equilibrium branch in Figure 8. The remaining radii are monotone in δ : decreasing for small $\delta > 0^\circ$ (from infinity at straight motion) and increasing for $85^\circ < \delta < 90^\circ$.

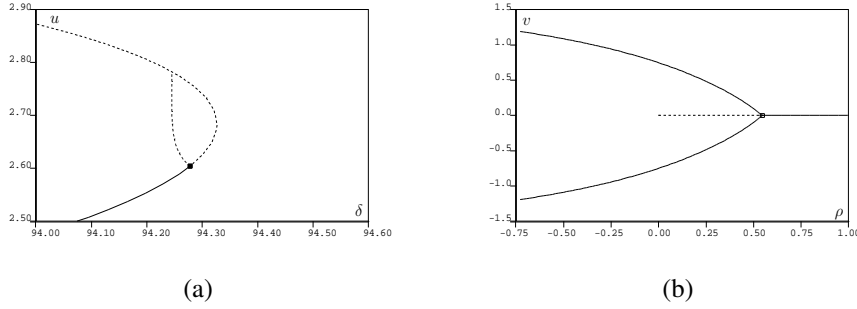


Figure 10: (a) Branch of periodic solutions emerging from a Hopf bifurcation. See Figure 8. It terminates in a homoclinic bifurcation. All periodic solutions are unstable. (b) Bifurcation diagram for changing rudder area $A_R^* = (1 + \rho)A_R$ with A_R from the Hamburg test case; note that $\delta = 0$ is fixed. The two stable branches emerging from the (degenerate) pitchfork bifurcation intersect $\rho = 0$ at the equilibria from the branch in Figure 8 at $\delta = 0$.

Part of the radii of the corresponding circular ship motion, given by $\sqrt{u^2 + v^2}/R$, are plotted in Figure 9. For smaller value of $\delta > 0$ the radii are monotone. Notably, this graph has two minima with essentially the same radii of $\approx 119\text{m}$ at quite different rudder angles δ . Hence, the smallest radius the ship of length 153m can make at equilibrium is roughly 75% its length. Note that the computations in Section 3.3 give the same results. See Figure 13.

Branch part II: Continuing further along the branch plotted in Figure 8, the u -components increase, and the equilibria undergo a Hopf bifurcation at $\delta \approx 94.28^\circ$, as well as another fold at $\delta \approx 94.32^\circ$. The branch of periodic solutions emerging at the Hopf bifurcation is plotted in Figure 10(a). It consists of unstable periodic solutions and the branch terminates in a homoclinic bifurcation, analogous to the cases discussed for the Thruster model.

3.2.2 Stability change of straight motion

Analogous to Section 2.2.3, where a mass parameter was changed in the Thruster model, here we change a parameter of the Hamburg test case set to stabilize straight motion in the rudder model. In this case we choose the rudder area A_R . As plotted in Figure 10(b), at roughly 1.5 times this rudder area, the straight motion becomes stable through a (degenerate) supercritical pitchfork bifurcation, much like in the Thruster model for the ship mass. A possible interpretation is that the intrinsic instability of the hull for straight motion can only be compensated by a sufficiently large rudder.

3.2.3 Other branches of equilibria: Sensitivity of stable motion

The numerical analysis with MATLAB presented in Section 3.3 shows existence of other equilibria than those connected to straight motion. Indeed, we can use AUTO to continue from these solution to reveal the entire branches, which are plotted in Figure 11. These branches form a symmetric pair of loops and all solutions on these

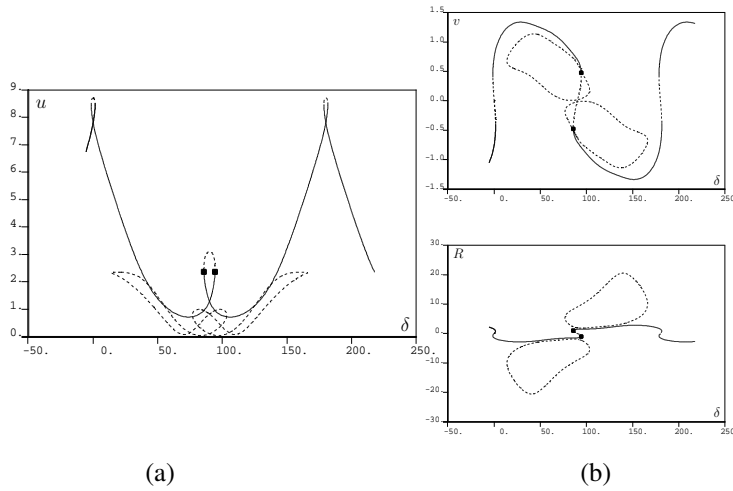


Figure 11: Bifurcation diagram of all known branches of equilibrium solutions. (a) The u -component, (b) the v -component (upper panel) and R -component (lower panel). In addition to Figure 8 here are two loops of unstable equilibria, symmetric about $\delta = 90^\circ$. These loops have symmetric counterparts by reflection about $\delta = 0$ that are not shown.

symmetric branches are unstable.

A potential relevance of these solutions is their proximity to the stable branches for δ between 60° and 140° . This suggests that the basin of attraction of the stable solutions becomes smaller for larger angles, that is, the stable motion is increasingly sensitive to perturbations. However, the basin of attraction, in particular for smaller δ , may be constrained by other nonlinear solutions that our analysis does not reveal.

3.3 Numerical Analysis of the Rudder Model using MATLAB

The results of Section 3.2 were computed using the program AUTO, which starts with one equilibrium solution of the equations of motion and from it computes a whole branch of solutions. We found the starting equilibrium solution for Figures 8–9 and 11 using MATLAB. In this section we describe in detail how the MATLAB program works. The MATLAB program not only finds equilibrium solutions to provide to AUTO, but also reproduces parts (but not all) of the bifurcation diagrams given in Section 3.2 and, like AUTO, determines whether or not the solutions are stable. We see a good agreement between the results of the two programs. In this section we present several eigenvalue figures. Note that AUTO also computes the eigenvalues, but we just do not show them in Section 3.2.

Model. Recall that the (dimensional) equations of motion of the ship (1) have the form

$$M\dot{\mathbf{u}} = \mathbf{f}(\mathbf{u}; \mathbf{p}), \quad (22)$$

where \mathbf{u} is a vector of velocities and \mathbf{M} is a matrix of mass and added mass coefficients:

$$\mathbf{u} = \begin{bmatrix} u(t) \\ v(t) \\ r(t) \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} m + m_{uu} & 0 & 0 \\ 0 & m + m_{vv} & m_{vr} \\ 0 & m_{rv} & I_z + m_{rr} \end{bmatrix}. \quad (23)$$

The vector-valued function \mathbf{f} is the nonlinear function of \mathbf{u} defined by the right-hand side of equation (1) and it represents the resultant force on the ship:

$$\mathbf{f}(\mathbf{u}; \mathbf{p}) = \begin{bmatrix} mrv + X_G \\ -mru + Y_G \\ N_G \end{bmatrix}. \quad (24)$$

Note that \mathbf{f} depends on the vector of control and design parameters \mathbf{p} . The control parameters are the rate of rotation of the propeller n and the rudder angle δ , and the design parameters, which are many, include the area, aspect ratio and position of the rudder, the diameter of the propeller, and the length and mass of the ship.

Stability theory. Suppose that the constant vector $\mathbf{u}_e = (u_e, v_e, r_e)^T$ is an equilibrium solution of (22) for parameter vector \mathbf{p} , i.e., $\mathbf{f}(\mathbf{u}_e; \mathbf{p}) = 0$. Recall that equilibrium solutions of (22) correspond to straight motions of the ship if the rudder angle $\delta = 0$ and $v = r = 0$, or to turning circles otherwise. Standard results in the theory of ordinary differential equations (see, e.g., [10, Chapter 3] or [11, Chapter 9]) state that the stability of the equilibrium solution \mathbf{u}_e is determined by the eigenvalues λ of the (generalised) eigenvalue problem

$$\mathbf{Df}(\mathbf{u}_e; \mathbf{p}) \mathbf{v} = \lambda \mathbf{M} \mathbf{v}, \quad (25)$$

where $\mathbf{Df}(\mathbf{u}_e; \mathbf{p})$ is the Jacobian matrix of \mathbf{f} evaluated at $(\mathbf{u}_e; \mathbf{p})$. It has components $[\mathbf{Df}]_{ij} = \partial f_i / \partial u_j$. The vector \mathbf{v} is an eigenvector. (The eigenvalue problem (25) can be derived by linearising (22) about the equilibrium solution \mathbf{u}_e and then seeking solutions with exponential time dependence $e^{\lambda t}$.) Recall that if all the eigenvalues λ of (25) have negative real part, then \mathbf{u}_e is asymptotically stable. If at least one of the eigenvalues has positive real part (and the other eigenvalues have nonzero real part), then \mathbf{u}_e is unstable.

Algorithm. We compute equilibrium solutions \mathbf{u}_e and the corresponding eigenvalues λ in MATLAB in the following way: Fix the parameter vector \mathbf{p} . Starting from an initial guess \mathbf{u}_0 , we solve the nonlinear algebraic equation $\mathbf{f}(\mathbf{u}_e; \mathbf{p}) = 0$ using the MATLAB function *fsolve*. The Jacobian matrix $\mathbf{Df}(\mathbf{u}_e; \mathbf{p})$ is then approximated using the centered difference formula. Finally, the eigenvalue problem (25) is solved using the MATLAB function *eig*. This computation takes a fraction of a second.

Parameter values. For our computations we took the design parameters of the ship to be fixed and equal to the values given in [7] for the Hamburg Test Case. For the wake fraction w and thrust deduction factor t , whose values are not given in [7], we took $w = 0.38$ and $t = 0.22$. For the control parameters, we took the rate of rotation of the propeller $n = 2$ Hz (number of rotations per second). We took the rudder angle δ to be the variable parameter and studied how the stability of equilibrium solutions depends on it.

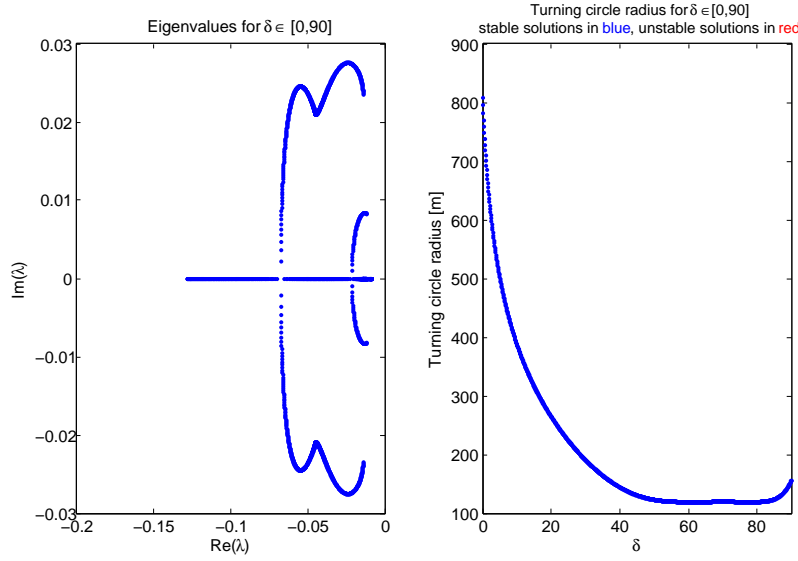


Figure 12: Results of Simulation 1. The graph on the left shows the eigenvalues corresponding to the equilibrium solutions for $\delta \in [0, 90]$ degrees (sampled every 0.1 degree). There are three eigenvalues per solution. Since all the eigenvalues lie in the left half-plane, all the equilibrium solutions are stable. The graph on the right shows the radius of the turning circle for each value of the rudder angle δ .

Results: Simulation 1, stable turning circles. Figures 12 and 13 show the results of the computation starting from initial guess $\mathbf{u}_0 \equiv (u_0, v_0, r_0) = (6 \text{ ms}^{-1}, 1 \text{ ms}^{-1}, 0 \text{ radians}\cdot\text{s}^{-1})$ for $\delta = 0$. We incremented δ in steps of 0.1 degrees from 0 to 90 degrees and every time we incremented δ we updated the initial guess \mathbf{u}_0 , taking it to be the value of the equilibrium solution computed for the previous value of δ , i.e., $\mathbf{u}_0(\delta_{i+1}) = \mathbf{u}_e(\delta_i)$. To compute equilibrium solutions \mathbf{u}_e and eigenvalues λ for the entire range of δ took around only 15 seconds. As seen from Figure 12, the program computes a family of *stable* turning circles (since all the eigenvalues lie in the left half-plane). The smallest turning circle has radius 118.94 m and is obtained at rudder angles $\delta = 60.4$ and $\delta = 78.4$ degrees. See Figure 13. Figure 13 agrees with the stable branch of the bifurcation diagram produced in AUTO (compare Figure 13 with Figures 8–9).

Results: Simulation 2, branch jumping. In this simulation we show the importance of the initial guess \mathbf{u}_0 in determining which branches of solutions the MATLAB program finds. This time we keep the initial guess fixed at $\mathbf{u}_0 \equiv (u_0, v_0, r_0) = (6 \text{ ms}^{-1}, 1 \text{ ms}^{-1}, 0 \text{ radians}\cdot\text{s}^{-1})$, the same initial guess that was used in Simulation 1 for $\delta = 0$, for all values of $\delta \in [0, 40]$ degrees; we do not update the initial guess when we update δ . The results are shown in Figures 14 and 15. Observe that for δ between 30 and 40 degrees, the program jumps between a stable and a nearby unstable branch of turning circle solutions. The stable branch is the same one that was produced in Simulation 1. We use a point on the unstable branch as the starting point

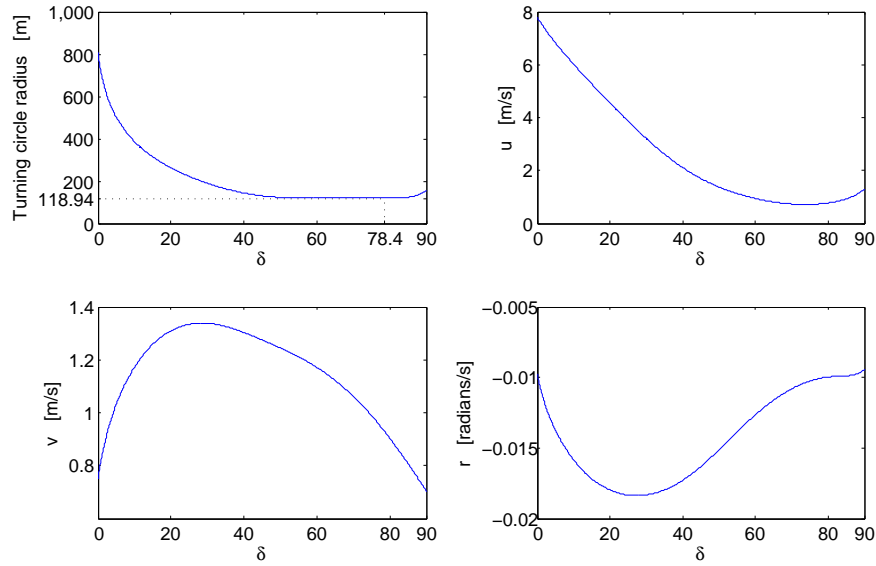


Figure 13: Results of Simulation 1. The turning circle radius and velocity components of the ship for rudder angles $\delta \in [0, 90]$.

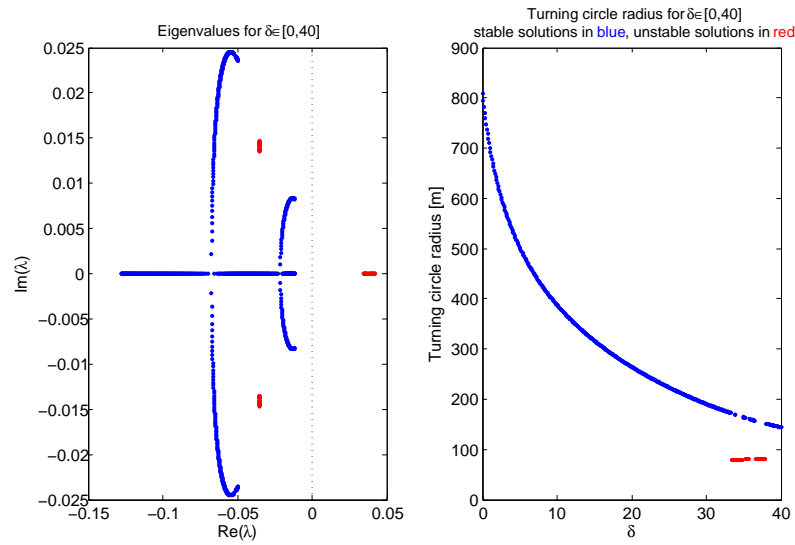


Figure 14: Results of Simulation 2. The graph on the left shows the eigenvalues corresponding to the equilibrium solutions for $\delta \in [0, 40]$ degrees. The nonlinear solver jumps between a stable and an unstable branch of solutions. The eigenvalues of stable solutions are blue and those of unstable solutions are red. The graph on the right shows the radius of the turning circle for each value of the rudder angle δ .

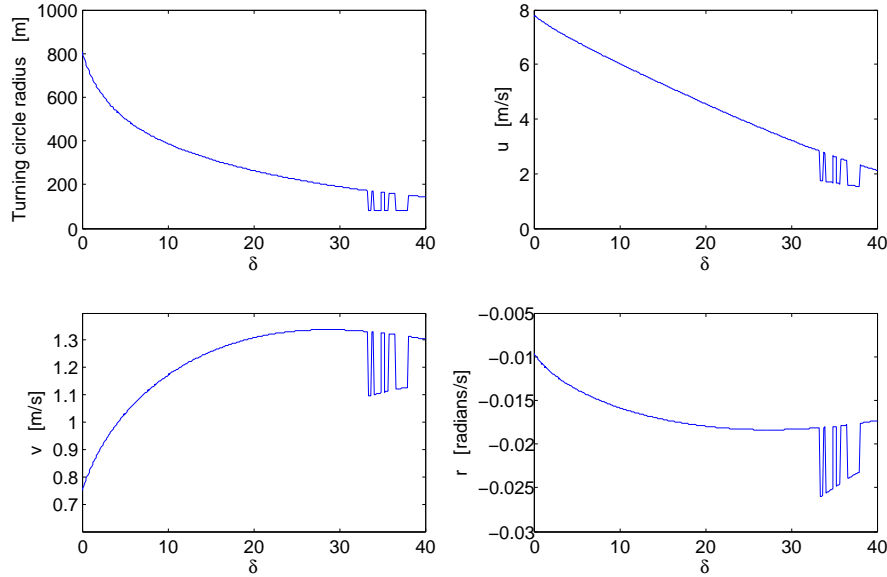


Figure 15: Results of Simulation 2. The discontinuities in the graphs are due to the nonlinear solver jumping between stable and unstable solution branches.

for Simulation 3.

Results: Simulation 3, unstable turning circles. In this simulation we produce an unstable branch of turning circle solutions for $\delta \in [37, 90]$ degrees. For $\delta = 37$ degrees, we take the initial guess \mathbf{u}_0 to be the solution computed in Simulation 2 for $\delta = 37$. We then update this initial guess every time that δ is incremented, as in Simulation 1. The results are shown in Figures 16 and 17. In this case we see that every equilibrium solution computed is an unstable turning circle. The smallest turning circle has radius 63.74 m, which is smaller than the smallest turning circle computed in Simulation 1. Since the solution here is unstable, however, it would be impossible for the ship to perform the smaller turning circle. The complete unstable branch, of which Figure 17 forms a part, was computed in AUTO. Compare Figure 17 to Figure 11.

Summary and Remarks. Our MATLAB program is a fast way of computing equilibrium solutions of the equations of motion, determining their stability, and finding the smallest possible turning circle that a ship can perform. This algorithm is far quicker than time-integrating the equations of motion. In the simulations above we chose to fix the design parameters of the ship and the propeller speed, and to vary the rudder angle. The program, however, allows any of the parameters to be varied. For example, the user could study the effect of varying the rudder area on the turning circle manoeuvre.

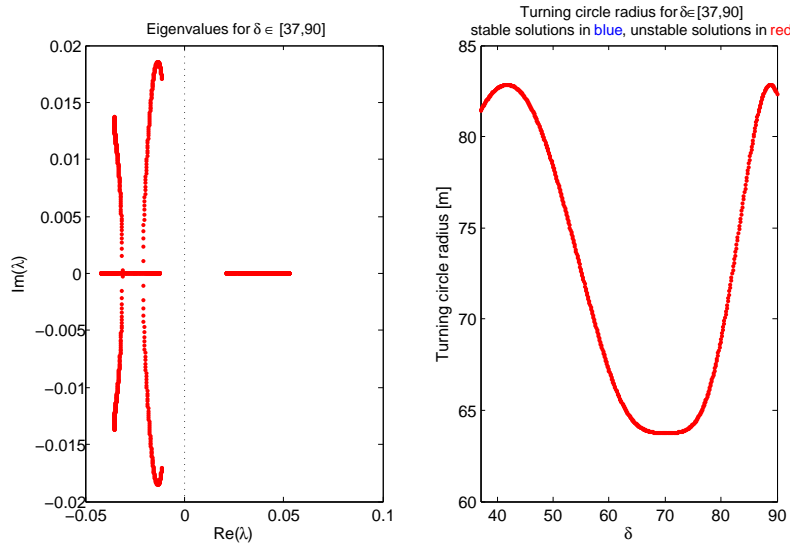


Figure 16: Results of Simulation 3. The graph on the left shows the eigenvalues corresponding to the equilibrium solutions for $\delta \in [37, 90]$ degrees. Every solution has one eigenvalue in the right half-plane and so all the solutions are unstable. The graph on the right shows the radius of the turning circle for each value of the rudder angle δ .

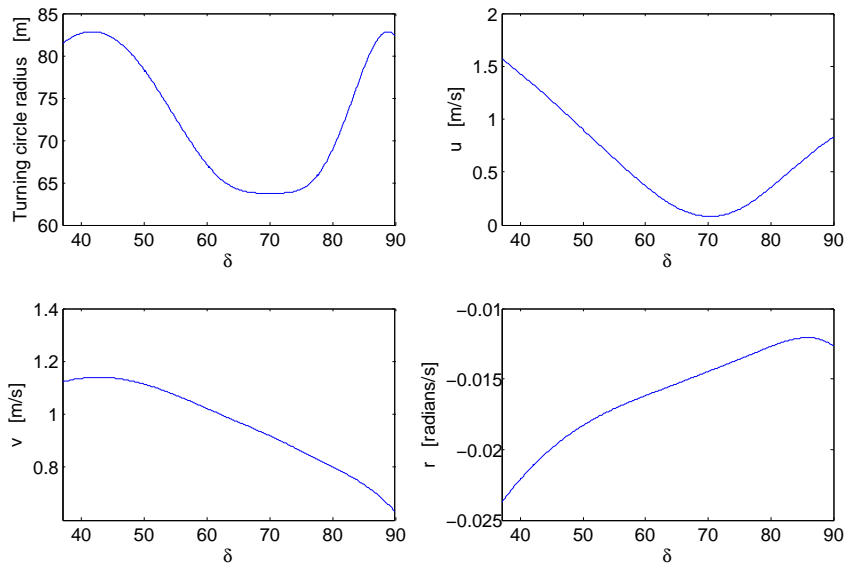


Figure 17: Results of Simulation 3. The turning circle radius and velocity components of the ship for rudder angles $\delta \in [37, 90]$. All these solutions are unstable.

4 Discussion & Outlook

We have numerically and theoretically analyzed the ‘Rudder model’ for ship manoeuvring provided by MARIN, and introduced a simplified ‘Thruster model’ that roughly combines rudder and propeller. We have shown that steady states in the model correspond to circular motion of the ship and computed the corresponding radii. We have non-dimensionalized the models and thereby remove a number of parameters, so that, due to a scaling symmetry, only the rudder (or thruster) angle remain as a free parameter.

Using ‘degree theory’, we have shown that a slight modification of the models possesses at least one steady state for each angle, and we have found certain constraints on the possible steady state configurations regarding stability.

We have shown that straight motion is unstable for the Hamburg test case and have used numerical continuation and bifurcation software to compute a number of curves of states together with their stability, and the corresponding radii of the ship motion. In particular, for the Rudder model straight forward motion can be stabilized by increasing the rudder size parameter, and the smallest possible radius is ~ 119 m. For the Thruster model we show that this also works when decreasing the ship weight parameter.

These analyses have illustrated methods and tools from nonlinear dynamical systems theory that can be used to analyse a model without simulation. Compared with simulations, the numerical bifurcation analysis is much less time consuming. We have implemented the model in MATLAB and provide a simplistic continuation method. The advanced continuation and numerical bifurcation analysis has been done with an implementation of the model in the software AUTO.

In conclusion, it is indeed possible to analyse the manoeuvring behavior of a ship, both in a qualitative sense and in a quantitative sense, based on the mathematical model alone, without the need for explicit solutions from by time integration methods.

As an outlook, we remark that it is possible to automatize much of the numerical computation in order to make these techniques available to non-experts. This could lead to a more generic design tool which allows the formal definition of the manoeuvring equations, the specification of the coefficients and the definition of the design input and output parameters (e.g. turning circle diameter vs rudder angle).

It is also possible to devise and analyse control techniques that stabilize a desired course of the ship. On the other hand, the analysis in this report does not cover the modified rudder forces for other courses than straight ahead.

A Appendix: The topological degree

In this appendix, we will give some more information on the topological degree and we will explain and prove Proposition 2.2.

To define the degree of 2.2 there are various possibilities. The best way is to use homology and cohomology theory. Unfortunately this requires (even) more machinery than we want to use here. Instead we use the *homotopy groups* which are comparatively easy to define and easy to compute for the case we need it, even though they are subtle and difficult to compute in general. We also use *differential forms*, which are very concrete objects whose calculus is a generalization of classical vector calculus but *much* easier to compute with, both for people and computers. The material in this section is standard but the presentation of 2.2.2 is a rehash of folklore and simplifications of more general constructions. A quick introduction to differential forms is in [3], [9] and in the nice little book [2]. Their relation to topology is in the equally nice [1], and to numerical mathematics (aka discrete exterior calculus) in [4]. More topology can be found in [8] and [6].

The n -th homotopy group $\pi_n(X)$ of a space X , is the set of *equivalence classes* of continuous maps $\sigma : S^n \rightarrow X$ of the n -sphere to X .² Two maps $\sigma_1, \sigma_2 : S^n \rightarrow X$ are *equivalent* if one can be deformed in the other with a 1 parameter family σ_s , a so called homotopy. The idea is that once everything that can be deformed by continuous deformations is equivalent, what is left is something discrete and (more) computable. Two maps σ_1 and σ_2 can be juxtaposed (in dimension 1 this is running one path after the other), and this defines a composition map on $\pi_n(X)$. The neutral element is the map that sends a sphere to a point in X and the inverse of the composition is given by precomposing with reflection in a plane (or equivalently, any orthogonal map with determinant -1 ; in dimension 1, this corresponds to running a path backwards). The upshot is that $\pi_n(X)$ is a group for $n \geq 1$ and an Abelian group for $n \geq 2$. Moreover for every continuous map $f : X \rightarrow Y$ there is a map $f_* : \pi_n(X) \rightarrow \pi_n(Y)$ by postcomposition i.e. given a map $\sigma : S^n \rightarrow X$ defining an equivalence class $[\sigma] \in \pi_n(X)$ we define $f_*[\sigma] \in \pi_n(Y)$ as the equivalence class of $f \circ \sigma : S^n \rightarrow X \rightarrow Y$. One easily checks this is well defined. It is also easy to check that $(f \circ g)_* = f_* \circ g_*$ (functoriality), that f_* is a group homomorphism and that f_* only depends on the homotopy class of f . We find in particular that if $i : X \hookrightarrow Y$ is a subspace and there is a projection $p : Y \rightarrow X$ which is homotopic to the identity, then $i_* : \pi_n(X) \xrightarrow{\cong} \pi_n(Y)$ is an isomorphism. In particular we see that

$$i_* : \pi_n(S^m) \cong \pi_n(\mathbb{R}^m - \{0\}) \quad (26)$$

To actually compute the homotopy groups is not at all easy, in fact it is an active research area to compute and understand the higher homotopy groups of simple spaces like the n -sphere. However, the following is a very well-known and basic theorem of algebraic topology,

$$\pi_k(S^n) \cong 0 \text{ for } 1 \leq k < n \quad (27)$$

$$\cong \mathbb{Z} \text{ for } k = n \quad (28)$$

²This is a slight oversimplification, we have to map base-points to base-points, for example to define addition. Here we can ignore this

Note that the identity map gives a canonical generator for the group $\pi_n(S^n)$.

We can now define the degree of a continuous vector field $F : S^{n-1} \rightarrow \mathbb{R}^n - \{0\}$. The vector field defines a group homomorphism

$$F_* : \pi_{n-1}(S^{n-1}) \cong \mathbb{Z} \rightarrow \pi_{n-1}(\mathbb{R}^n - \{0\}) \cong \pi_{n-1}(S^{n-1}) = \mathbb{Z}.$$

Group homomorphism $\phi : \mathbb{Z} \rightarrow \mathbb{Z}$ are very concrete things because they are just multiplication by the integer $\deg(\phi) = \phi(1)$. For example

$$\phi(3) = \phi(1 + 1 + 1) = \phi(1) + \phi(1) + \phi(1) = 3\phi(1) = 3\deg(\phi).$$

We define

$$\deg(F) = \deg(F_*).$$

We now turn to the calculus of differential forms. An elementary differential form of dimension k on an open set $U \subset \mathbb{R}^n$ is a formal expression of the form

$$dx^{i_1} \wedge dx^{i_2} \wedge \cdots \wedge dx^{i_k}$$

a differential form ω is a sum of elementary differential forms with function coefficients i.e.

$$\omega = \sum_{i_1, \dots, i_k} \omega_{i_1, \dots, i_k}(x_1, \dots, x_k) dx^{i_1} \wedge dx^{i_2} \wedge \cdots \wedge dx^{i_k}$$

In the following we will assume the function coefficients $\omega_{i_1, \dots, i_k}(x_1, \dots, x_k)$ to be smooth. The wedge product \wedge in an elementary differential form is associative but it anti-commutes:

$$dx^i \wedge dx^j = -dx^j \wedge dx^i.$$

We can thus assume that the indices are all different. If the set $\{i_1, \dots, i_k\} \neq \{j_1, \dots, j_k\}$, then $dx^{i_1} \wedge \cdots \wedge dx^{i_k}$ is independent of $dx^{j_1} \wedge \cdots \wedge dx^{j_k}$. The interpretation is that differential forms keep track of what flows through an infinitesimal surface or volume element in a certain direction (of which there can be many). For example the 2 form

$$j = j_{12}dx \wedge dy + j_{23}dy \wedge dz + j_{31}dz \wedge dx \quad (29)$$

has a natural physical interpretation as a current with e.g. $j_{23}dy \wedge dz$ the amount of mass (or charge or energy...) per unit time flowing through an infinitesimal part of the oriented y - z plane and it correspondingly has units of kg/sec (or Cb/sec, or J/sec ...) rather than kg/(sec m²).

We can *pull back* a differentiable form with a smooth map $f : V \rightarrow U$, where $V \subset \mathbb{R}^m$ to get a new k form $f^*\omega$ on V . What this means is that if we write the map f in coordinates as

$$f(y_1, \dots, y_m) = (x^1(y_1, \dots, y_m), \dots, x^n(y_1, \dots, y_m))$$

then we use linearization to expand $dx^i = \sum_j \frac{\partial x^i}{\partial y^j} dy^j$ and multiply everything out using the anti-commutativity of the wedge product. What we get is an expression which “simplifies” in a expression in determinants of minors of the Jacobian and elementary differential forms $dy^{j_1} \wedge \cdots \wedge dy^{j_k}$ with $j_1 < \cdots < j_k$. It is easy to see that

if $g : W \rightarrow V$ is another smooth map, then $(f \circ g)^* = g^* \circ f^*$ (another instance of functoriality).

What makes this useful is that forms can be integrated over an oriented smoothly embedded k -simplex $\sigma : \Delta_k \rightarrow U$.

$$\int_{\sigma} \omega = \int_{\Delta_k} \sigma^* \omega = \int_{\Delta_k} \sum \omega_{i_1, \dots, i_k} \det \left(\frac{\partial(x^{i_1}, \dots, x^{i_k})}{\partial(t^1, \dots, t^k)} \right) dt^1 \wedge \dots \wedge dt^k$$

where the last expression can be interpreted using normal Riemann or Lebesgue integration. The integral is independent of the curvilinear coordinates we use, because the forms keep track of all the necessary Jacobians, which is the point of introducing them. We can therefore extend the definition of integration of a k form over smoothly embedded oriented k dimensional compact sub-manifolds $\sigma : K \rightarrow U$, by triangulation.

The other thing to know about differential forms is Stokes's formula, a better behaved, and more user friendly generalization of the Gauss and Stokes formulas. Define the exterior derivative d as

$$d\omega = \sum_{j, i_1, \dots, i_k} \frac{\partial \omega_{i_1, \dots, i_k}}{\partial x^j} dx^j \wedge dx^{i_1} \wedge \dots \wedge dx^{i_k}.$$

It commutes with pullback i.e. $d(f^* \omega) = f^* d\omega$. In particular, with the exterior derivative we can change curvilinear coordinates at will, unlike the classical divergence and curl which work different in rectangular and spherical coordinates. The exterior derivative therefore works fine on smooth manifolds. Now for a piecewise smooth k -dimensional manifold M with boundary ∂M we have

$$\int_{\partial M} \omega = \int_M d\omega \quad (30)$$

For example for the 2-form current j of (29), and $M \subset \mathbb{R}^3$, equation (30) boils down to

$$\int_{\partial M} j = \int_M (\partial_z j_{12} + \partial_x j_{23} + \partial_y j_{31}) dx \wedge dy \wedge dz$$

in which we recognize the classical Gauss formula.

It follows from Stokes formula that if $\sigma : S^{n-1} \rightarrow \mathbb{R}^n - \{0\}$ and $\sigma' : S^{n-1} \rightarrow \mathbb{R}^n - \{0\}$ are smoothly homotopic smooth maps and ω is a form on $\mathbb{R}^n - 0$ with $d\omega = 0$ (a so called closed form), then

$$\int_{\sigma} \omega = \int_{\sigma'} \omega$$

One can show that the homotopy classes of spheres $\pi_n(S^n)$ are represented by smooth maps and that for smooth maps, equivalence up to smooth homotopies is the same as equivalence up to continuous homotopies. Thus it makes sense to integrate over homotopy classes and it is then easy to see that $\int_{k[\sigma]} \omega = k \int_{\sigma} \omega$. This will allow us to compute degrees and prove the local to global degree formula 2.2.2³

³This construction is a version of the de Rham cohomology construction of the localised Euler class. See [1].

For notational simplicity we set $n = 3$. Choose a form ω written in spherical coordinates on $\mathbb{R}^3 - \{0\}$ as $\omega = f(\phi, \theta)d\phi \wedge d\theta$ such that

$$\int_{S^2} \omega = 1.$$

For example we could take $f(\phi, \theta) = -\sin(\theta)$, but we could also take f to be a bump function with support near $(\phi, \theta) = (0, \pi/2)$, and it is interesting to compare the interpretation of the results below. Clearly $d\omega = 0$ because there is no r -derivative and, say, the θ -derivative comes with a $d\theta$, but $d\theta \wedge d\theta = 0$. Now for a smooth vector field $F : S^2 \rightarrow \mathbb{R}^3 - \{0\}$, we have

$$\deg(F) = \deg(F) \int_{S^2 \subset \mathbb{R}^3 - \{0\}} \omega = \int_{F_*(S^2(V_0))} \omega = \int_{S^2(V_0)} F^* \omega$$

Suppose further that F is continuously differentiable and non-vanishing on $\Omega - Z_\epsilon$ where Z_ϵ is an ϵ neighborhood of the zeroes of X with smooth boundary and $\partial\Omega = S^2$.

$$\begin{aligned} \deg(F) &= \int_{S^2} F^* \omega \\ &= \int_{\partial(\Omega - Z_\epsilon)} F^* \omega + \int_{\partial Z_\epsilon} F^* \omega \\ &= \int_{\Omega - Z_\epsilon} F^* \underbrace{d\omega}_0 + \int_{\partial Z_\epsilon} F^* \omega \\ &= \int_{\partial Z_\epsilon} F^* \omega \end{aligned}$$

Now finally assume that F is continuously differentiable on Ω and transversal with isolated zeros in z_1, \dots, z_m . Then ∂Z_ϵ is a union of small 2-spheres $S_1^2(\epsilon), \dots, S_m^2(\epsilon)$. If the 2-spheres $S_i^2(\epsilon)$ are chosen small enough, then on $S_i^2(\epsilon)$ $F(x) = L_i(x) + o(\epsilon)$ where $L_i(x) = (DF)_{z_i}(x - z_i)$ is the linearization of F around z_i . The Jacobian $(DF)_{z_i}$ is invertible (by definition of transversality), so by compactness of $S^2(\epsilon)$, $|L_i(x)| \geq c\epsilon$. Hence, the rest term can be estimated on $S^2(\epsilon)$ as $o(\epsilon) < 1/2|L_i(x)|$ for ϵ sufficiently small. We can therefore make a linear homotopy of the restriction $F|_{S_i^2(\epsilon)}$ to the linearization $L_i|_{S_i^2(\epsilon)}$ similar to (20). We conclude that

$$\begin{aligned} \deg(F) &= \int_{\partial Z_\epsilon} F^* \omega \\ &= \sum_i \int_{S_i^2(\epsilon)} F^* \omega \\ &= \sum_i \int_{F_*(S_i^2(\epsilon))} \omega \\ &= \sum_i \int_{L_i^*(S_i^2(\epsilon))} \omega \end{aligned}$$

But the degree of a linear vector field $x \rightarrow Ax$ with A invertible, depends only on the sign of $\det(A)$ i.e. the connected component of $\mathrm{GL}(n, \mathbb{R})$ containing A . We finally conclude that

$$\deg(F) = \sum_i \mathrm{sign}(\det((DF)_{z_i})). \quad (31)$$

References

- [1] R. Bott and L.W. Tu. *Differential forms in algebraic topology*. Springer, 1982.
- [2] T. Bröcker and K. Jänich. *Introduction to differential topology*. Cambridge University Press, 1982.
- [3] Y. Choquet-Bruhat, C. DeWitt-Morette, and M. Dillard-Bleick. *Analysis, manifolds, and physics*. North-Holland, 1982.
- [4] M. Desbrun, A.N. Hirani, M. Leok, and J.E. Marsden. Discrete exterior calculus. *Arxiv preprint math/0508341*, 2005.
- [5] E. Doedel, R.C. Paffenroth, A.R. Champneys, T.F. Fairgrieve, Y.A. Kuznetsov, B.E. Oldeman, B. Sandstede, and X. Wang. Auto2000: Continuation and bifurcation software for ordinary differential equations (with homcont). *Technical report, Concordia University*, 2002.
- [6] A. Hatcher. *Algebraic topology*. Cambridge Univ Press, 2002.
- [7] MARIN. *Studiegroep Wiskunde met de Industrie 2011. MARIN - Manoeuvring Behaviour of Ships. Report No.: SWI-MARIN*. January 2011.
- [8] J.R. Munkres. *Elements of algebraic topology*. Westview Press, 1984.
- [9] R. Sjamaar. Manifolds and Differential forms. *Lecture Notes, Cornell University*, 2001.
- [10] C.J. van Duijn and M.J. de Neef. *Analyse van Differentiaalvergelijkingen*. VSSD, 2001.
- [11] Boyce W.E. and DiPrima R.C. *Elementary Differential Equations*. Wiley, 8th edition, 2004.

Routing for analog chip designs at NXP Semiconductors

*Marjan van den Akker** *Theo Beelen[†]* *Rob H. Bisseling**
*Bas Fagginger Auer** *Frederik von Heymann[‡]* *Tobias Müller[§]*
Joost Rommes[†]

Abstract

During the study week 2011 we worked on the question of how to automate certain aspects of the design of analog chips. Here we focused on the task of connecting different blocks with electrical wiring, which is particularly tedious to do by hand. For digital chips there is a wealth of research available for this, as in this situation the amount of blocks makes it hopeless to do the design by hand. Hence, we set our task to finding solutions that are based on the previous research, as well as being tailored to the specific setting given by NXP.

This resulted in an heuristic approach, which we presented at the end of the week in the form of a prototype tool. In this report we give a detailed account of the ideas we used, and describe possibilities to extend the approach.

1 Introduction

1.1 NXP Semiconductors

NXP Semiconductors N.V. (Nasdaq: NXPI) is a global semiconductor company and a long-standing supplier in the industry, with over 50 years of innovation and operating history. The company provides high-performance electronic chips to its customers, and produces these building on its expertise in the areas of RF, analog and digital circuits, power management, and security. These innovations are used in a wide range of automotive, identification, wireless infrastructure, lighting, industrial, mobile, consumer and computing applications. Headquartered in Europe, the company has approximately 28,000 employees working in more than 25 countries and posted sales of USD 3.8 billion in 2009.

1.2 Place and route for analog designs

The increasing demand for smaller, faster, and multi-functional electronic devices such as smart phones is one of the driving forces in the semiconductor industry. Combined with requirements on power usage, sustainability, and wireless functionality this

*Utrecht University

[†]NXP Semiconductors Netherlands N.V.

[‡]Delft University of Technology

[§]Centrum Wiskunde & Informatica, Amsterdam

is generating challenges in several domains. During the design of the layout of a chip, which is a representation of the chip in shapes in its physical layers (silicon, oxide, metal), one of the challenges is to place and route (wire) the circuit components in an optimal way. Aspects that define optimality may vary per design/application and are typically related to the area (convex hull) of the chip, the total wire length, and the unintended side-effects caused by the wires (crosstalk, i.e., electrical fields between wires). The place and route problem is further complicated by design rules and geometrical constraints.

The place and route problem has been studied for many years and mature solutions are available for digital designs [1, 4, 10, 11, 12, 15, 16, 17, 21], which typically consist of (almost) equally sized blocks and predefined routing channels. For analog designs, the main interest of NXP, the place and route problem is more complicated because blocks can vary in size and aspect ratio and may even overlap (so that there is no need for explicit routing), and because routing channels are typically not available but defined by the placement. Also, while typically several layers (metal) are available to route in, often one would like to limit this to as few layers as possible, and in some cases routing is even restricted to a single layer. Other objectives one can think of are to minimize the wire length and the number of turns in the wires, and typical constraints are that wires are either horizontal or vertical (only 90-degree turns) and should be at a certain minimum distance from each other. Furthermore, it is desirable to have a routing algorithm that is robust with respect to (small) changes in the layout, so that it can be used in parametrized designs to update the routing automatically when parameters change. This allows designers to quickly explore different physical design variants.

The challenge NXP set for the study week of 2011 was to develop an algorithm that, given a number of circuit blocks and their interconnections, computes an optimal layout including placement and routing.

1.3 Outline

In Section 2 we describe the precise task we discussed during the study week, and we give a detailed account of the partial results we were able to achieve, including the prototype of a tool that we think can simplify the work of chip designers. In Section 3 we give an overview of possible extensions and improvements of one particular aspect of our algorithm, which we believe could be a computational bottleneck for larger instances; Section 4 outlines a slightly more sophisticated algorithm than the one in Section 2, which also has the benefit of giving us lower bounds on the quality of the solutions it produces. We conclude the report with a summary of our results and an estimation of the success in terms of the original challenge.

2 Heuristic Method and Implementation

As illustration of the ideas developed for NXP during the study week, we implemented a heuristic for solving the problem as a C++ program. An early version of this program was demonstrated during the final presentation session of the study week as well as a later version during a visit to NXP in Eindhoven (Figure 1).

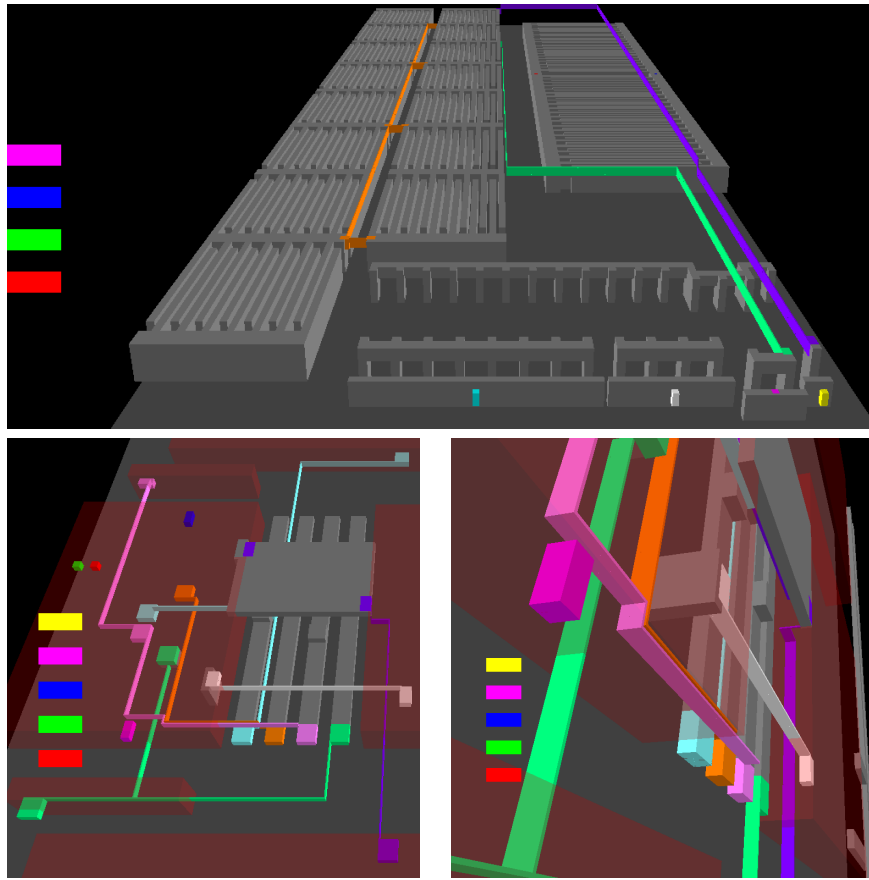


Figure 1: Screenshots from the demonstration at NXP, Eindhoven. The top picture shows generated routing for the `tunedCap` circuit and the bottom two pictures show routing for `difInvStage`. Both these circuits consist of 5 layers.

Because of the time constraints of the study week we chose to focus only on the routing aspect of NXP's problem: the program should be able to import a layout specified in NXP's circuit design software (Figure 2), where all the components have already been placed on the chip, and export routing in the form of wires connecting these components (thin wires in Figure 1). Wires are formed on the circuit by depositing metal in the production process.

The circuits are produced layer-by-layer, so we know beforehand that there is an $l \in \mathbb{N}$ specifying the number of layers ($l = 5$ in Figure 1) and a bounding box of the entire circuit board within which all components and wiring should be contained. We do not want to short-circuit different components on the circuit board by placing metal at the wrong places, therefore we receive for each layer a number of solid rectangles, in which no metal can be deposited (solid gray blocks in Figure 1). These rectangles are characterized by their lower-left and upper-right corners in \mathbb{R}^2 , as well as the number of the layer in which they are present. The components in the circuit need to

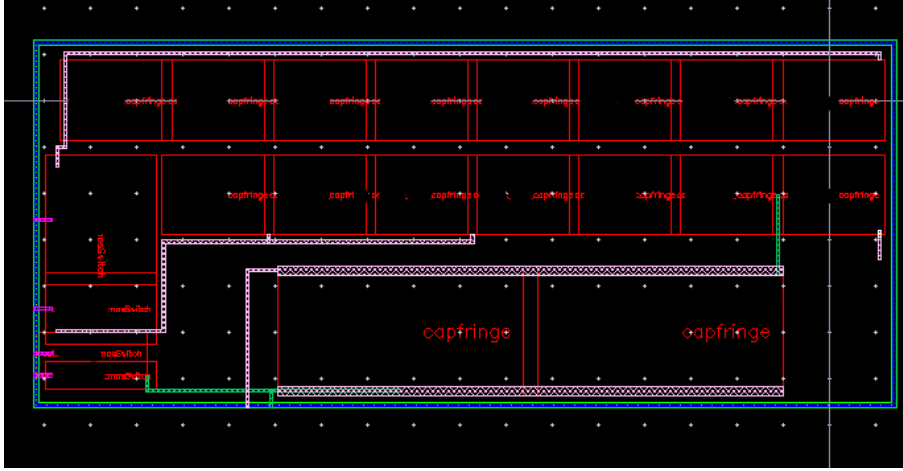


Figure 2: Circuit design software of NXP showing the layout of *tunedCap*.

be connected by conducting metal: we are given a list of *pins*, conducting rectangles belonging to a certain layer and having a certain color (the colored blocks in Figure 1). All pins sharing the same color should be connected by metal. To prevent accidental short-circuiting and interference, wires should have a minimum distance from each other; these minima can be different for different layers, so they are represented by a vector $\delta = (\delta_1, \dots, \delta_l) \in \mathbb{R}_{>0}^l$. It is important to note that wires are not restricted to a single layer: wires can make jumps to other layers by *vias*.

Our program now needs to find out, given these parameters, where to deposit metal in the circuit such that

- for each pair of pins sharing the same color there exists a continuous metal path connecting the pins;
- the distance between two bits of deposited metal in the same layer k that are connecting pins with different colors is always $\geq \delta_k$;
- no metal is deposited in the solid areas and no metal is deposited outside of the circuit board.

These are hard constraints in the sense that any solution which does not satisfy all these criteria is unacceptable.

2.1 Optimization

If we only took the hard constraints into account, we could end up with very unfavorable and costly solutions (e.g., paths that needlessly use a lot of metal). Therefore we will, in addition to satisfying the hard constraints, try to minimize the following quantities:

- the total amount of metal that needs to be deposited (as depositing metal costs money and long paths increase the resistance, which increases power consumption);

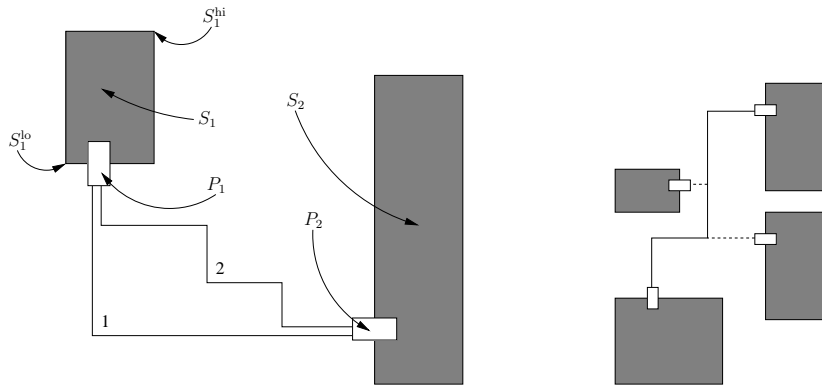


Figure 3: Left: two wires that use the same amount of metal, but turn a different number of corners. The blocks S_1 and S_2 are solid area bounding boxes, P_1 and P_2 are pin bounding boxes, and S_1^{lo} and S_1^{hi} are the lower-left and upper-right corners of S_1 . Right: Construction of a Steiner tree (see Section 3) by first creating the shortest path between the pins farthest away from each other (solid line) and then connecting the remaining pins to this path (dashed line).

- the number of corners in each of the paths (as corners introduce interference and are more susceptible to production errors, compare paths 1 and 2 in Figure 3);
- the number of jumps between different layers (as producing vias is expensive).

Depending on the specific circuit for which a routing needs to be generated, there could be additional objectives, such as

- limiting the amount of metal deposited in a specific layer;
- discouraging paths from lying on top of each other;
- encouraging metal deposition in certain areas of a certain layer, while discouraging it in others;
- ...

Hence the optimization criteria should be as customizable and flexible for the circuit designers as possible.

The heuristic (Algorithm 12) we use to solve the problem described above is based primarily on Dijkstra's shortest path finding algorithm [6] (in our implementation we improved performance by using the A* algorithm [9]; compare [11] and, for possible improvements, [17]). Dijkstra's algorithm is particularly useful for incorporating flexible optimization goals by viewing the minimization of total distance as the minimization of a more abstract cost function in which the goals of the chip designers are incorporated. Hence we look for *cheapest paths* with respect to this cost function (an example of which is given in Algorithm 13), using Dijkstra's algorithm.

2.2 Dijkstra's algorithm

Dijkstra's algorithm [6] computes the shortest paths between a single source vertex s and all vertices v in a directed graph $G = (V, E)$ with non-negative weights on the edges representing distances or more general costs; for our routing problem, the weights represent costs of various types. The algorithm works as follows. For each vertex v , a temporary best distance $d(v)$ is maintained. The distance $d(v)$ is lowered each time a shorter path to v is encountered. The predecessor of v in that path is then registered, thus storing the shortest path, and not only its cost. At some stage in the computation, the distance reaches its final value, the shortest distance to v . Then, v is included in the set D of finished vertices. Initially, $d(s) = 0$, $d(v) = \infty$ for $v \neq s$, and $D = \emptyset$. It can be shown that the distance attains its final value once $d(v) = \min_{w \in V-D} d(w)$. Algorithm 11 presents Dijkstra's algorithm.

Algorithm 11 Dijkstra's Algorithm.

Input: Directed graph (V, E) with costs $c(v, w) \geq 0$ for every edge $(v, w) \in E$, source vertex $s \in V$.

Output: The cost $d(v)$ for reaching vertex v from s , and a predecessor in a shortest path to v , for every $v \in V$.

```

1: for all  $v \in V$  do
2:    $d(v) \leftarrow \infty$ ;
3:    $pred(v) \leftarrow v$ ;
4:  $d(s) \leftarrow 0$ ;
5:  $D \leftarrow \emptyset$ ;
6: while  $D \neq V$  do
7:    $v \leftarrow \operatorname{argmin}\{d(w) : w \in V - D\}$ ;
8:    $D \leftarrow D \cup \{v\}$ ;
9:   for all  $w$  with  $(v, w) \in E$  do
10:    if  $d(v) + c(v, w) < d(w)$  then
11:       $d(w) \leftarrow d(v) + c(v, w)$ ;
12:       $pred(w) \leftarrow v$ ;
```

The A* algorithm is a more efficient variant of Dijkstra's algorithm, which makes use of knowledge about a target vertex t that we want to reach, such as a lower bound $l(v)$ on the cost of reaching t . In our case, we will use the minimum distance to be covered on a three dimensional grid (see the next section) as a lower bound, ignoring other costs. The bound is called *consistent* if $l(v) \leq c(v, w) + l(w)$ for all $(v, w) \in E$. For our problem, the bound is consistent because $c(v, w)$ includes the distance cost of routing from v to w . Algorithm 11 can be changed from Dijkstra to A* by writing $t \notin D$ instead of $D \neq V$ on line 6 and $d(w) + l(w)$ instead of $d(w)$ on line 7.

2.3 Implementation

Algorithm 12 outlines the heuristic employed by the prototype tool. In the algorithm we deposit metal of certain colors in order to be able to differentiate between wires connecting different sets of pins. Two deposited wires that do not share the same color should always be separated by the minimum separation distance as specified

by δ . For bounding boxes we use the following notation: if B is a bounding box, then $B^{\text{lo}}, B^{\text{hi}} \in \mathbb{R}^2$ are the lower-left and upper-right corner of the bounding box, respectively. The layer where the bounding box is present is indicated by $B^{\text{layer}} \in \mathbb{N}$.

Algorithm 12 Discretization and Heuristic Wiring.

Input: Number of layers $l \in \mathbb{N}$, minimum wire separation $\delta \in \mathbb{R}_{>0}^l$, circuit board bounding box B , solid area bounding boxes S_1, S_2, \dots , pin bounding boxes P_1, P_2, \dots .

Output: Discretized grid in which the metal depositions have been marked.

- 1: Let $\rho \leftarrow$ grid spacing based on δ or a specified resolution;
 - 2: Create three-dimensional grid G of $\lceil (B^{\text{hi}} - B^{\text{lo}})/\rho \rceil$ by l cells;
 - 3: **for all** solid area bounding boxes S **do**
 - 4: Mark all cells lying between $\lfloor (S^{\text{lo}} - B^{\text{lo}})/\rho \rfloor$ and $\lceil (S^{\text{hi}} - B^{\text{lo}})/\rho \rceil$ in layer S^{layer} as inaccessible for all colors;
 - 5: **for all** pins P **do**
 - 6: Mark all cells lying between $\lfloor (P^{\text{lo}} - B^{\text{lo}})/\rho \rfloor$ and $\lceil (P^{\text{hi}} - B^{\text{lo}})/\rho \rceil$ in layer P^{layer} as metal with color P^{color} ;
 - 7: Mark all cells within distance $\lceil \delta_{P^{\text{layer}}}/\rho \rceil$ of P as inaccessible, except for color P^{color} ;
 - 8: Sort pins by their color into nets and then the nets by increasing bounding box volume;
 - 9: **for all** nets \mathcal{P} in which all pins share the same color $\mathcal{P}^{\text{color}}$ **do**
 - 10: Find $P_1, P_2 \in \mathcal{P}$ such that the distance between the centers of P_1 and P_2 is maximal;
 - 11: Create a cheapest path $T \subseteq G$ from P_1 to P_2 traversing only cells accessible for color $\mathcal{P}^{\text{color}}$;
 - 12: Similarly create cheapest paths from T to all $P \in \mathcal{P}$ not connected to T and add these paths to T ;
 - 13: **for all** cells $c \in T$ **do**
 - 14: Mark c as metal with color $\mathcal{P}^{\text{color}}$;
 - 15: Mark all cells in layer c^z with distance $\leq \lceil \delta_{c^z}/\rho \rceil$ to c as inaccessible, except for color $\mathcal{P}^{\text{color}}$;
 - 16: **Output** G ;
-

To simplify the problem we first discretize it (line 2) to a regular three-dimensional grid G , with l layers and spacing ρ within each layer. G will be the graph in which we perform Dijkstra's path-finding algorithm. Cells $c \in G$ have three coordinates $(c^x, c^y, c^z) \in \mathbb{Z}^3$, where $1 \leq c^z \leq l$ is the layer in which the cell resides. To ensure that we never deposit metal in solid regions, we mark these in G first by discretizing the bounding boxes and flagging the cells contained in them as inaccessible for metal from any pin. We then proceed at line 5 to add all pins, marking the cells contained in them as metal of the pin's color and ensuring that no metal belonging to pins with a different color can be deposited near the pin (as this could violate the minimum separation criterion).

After the solid regions and pins have been marked, we generate paths connecting all the pins sharing the same color at line 8. First we cluster the pins together such that we have *nets* of pins all sharing the same color. Note that the algorithm will yield

different results if we connect the pins contained in these nets in a different order, hence we will fix the ordering by sorting the nets of pins of the same color by the volume of the bounding box containing the pins. This idea originates from the fact that connecting all the short paths first will lead to less conflicts between paths later than first connecting all the long paths (which could potentially cut off short paths).

Then we will connect, for each net of pins sharing the same color, the pins belonging to this net. If there are at most two pins in a net, we can directly use Dijkstra's algorithm to find the cheapest path connecting them. However, if we have more than two pins, we need to generate a *Steiner tree* (see Section 3) connecting all of them. In our heuristic, this is done by first creating a path between the two pins that are farthest apart, and then using this long path as a 'trunk' to which the remaining, unconnected, pins connect as branches via Dijkstra's algorithm (see the right panel of Figure 3). A path can only be created along cells that are accessible for the particular color of the current path; this to ensure that the minimum separation distance is always maintained.

Algorithm 13 Routing Cost Function for Dijkstra's Algorithm.

Input: Neighboring cells c_{-1} , c_0 , and c_1 in the grid G where c_{-1} is the predecessor of c_0 .

Output: The cost k to use cell c_1 to continue the path.

- 1: Initialize $k \leftarrow 0$;
 - 2: **if** c_1 is not marked as metal **then**
 - 3: We need to deposit metal: $k \leftarrow k + k^{\text{metal}}$;
 - 4: **if** $c_1^z \neq c_0^z$ **then**
 - 5: We need to create a via: $k \leftarrow k + k^{\text{via}}$;
 - 6: **if** $\|c_1 - c_{-1}\|^2 = 2$ **then**
 - 7: We turn a corner if we continue this way: $k \leftarrow k + k^{\text{corner}}$;
 - 8: ...
 - 9: Output k ;
-

Algorithm 13 gives a simple example cost function for Dijkstra's algorithm where we consider continuing an existing path going through cells $\dots \rightarrow c_{-1} \rightarrow c_0$ to c_0 's neighbor c_1 (neighbor in the sense that $\|c_1 - c_0\| = 1$). The cost can be influenced by varying three parameters: k^{metal} , k^{corner} , and k^{via} . This allows the designer to indicate whether he finds minimizing the length of the wires (increasing k^{metal}), minimizing the number of corners (increasing k^{corner}), or minimizing the number of vias (increasing k^{via}) more important. In Figure 1 the colored bars on the left are similar cost modifiers, from bottom to top: cost to deposit metal, cost to turn a corner, cost to create a via, cost to run over an existing wire, and cost to run over a solid block. By extending the cost function and permitting the designers to vary the associated weights, a number of different routing suggestions is easily obtained from the program. Note that the multiplicative factors such as k^{metal} can also be made to depend on the position of c_0 or c_1 , permitting the designer to make certain layers or certain regions of layers more or less attractive for the wires to traverse; this and other costs can be added at line 8.

The program prototype has been demonstrated to circuit designers of NXP in Austria and its source code has been provided to NXP.

3 Some Ideas for Steiner Trees

As we mentioned above, if we have more than two pins in one net, we cannot just use Dijkstra's algorithm to connect them. If we consider the pure problem of only one set of pins to be connected, this is an instance of the *Steiner tree problem*. Given a set of vertices, called terminals (which will be our pins), and another set of vertices, called Steiner points (grid points that are not blocked), a *Steiner tree* is any tree that contains all terminals and (some) Steiner points.

This section will give a brief account of possibilities to find a good Steiner tree in a grid (with obstacles). In particular this means that here we ignore the fact that it might not be possible (depending on the layout) to achieve an optimal routing solution by considering the nets successively. We defer this discussion to the next section.

As is true for most aspects of chip design, it is a computationally hard problem to find a smallest Steiner tree [7], and there is an abundance of strategies to get reasonably good approximations in acceptable time [2, 18]; and for more algorithms, see [10, 11, 12, 15]. Here we will restrict our attention to approaches that seem appealing because of their simple implementability and their compatibility with the strategy we chose for paths.

Most algorithms we found in the literature deal with the rectilinear version of the Steiner tree problem, i.e., where all terminals and Steiner points are given in a two-dimensional rectilinear grid. As we want to find Steiner trees in a three-dimensional grid (with obstacles), these results are to be taken with some caution, although we believe that on average they are close to what is to be expected for our setup. The idea presented in the previous section can be seen as a simplification of ideas from [10], where it is stated that we will get a tree that is at most a factor of $3/2$ away from a minimal Steiner tree, and on average much closer to it.

It should, however, be mentioned that the authors in [10] consider nets that include a source. In such a case it is usually desirable to minimize the distance of the other pins to the source, whereas our focus is more on the total (weighted) wire-length used in the tree. In the former case one typically gets star-shaped trees, whereas in the latter a caterpillar-structure is more likely.

As long as the net contains at most 6 pins, a *rectilinear* Steiner minimal tree in an obstacle-free grid can be found by going through all permutations of the order of pins, connecting them in these orders in the way described in Section 2.3. For larger nets, this approach will not always produce an optimal tree (not even in this special case of obstacle-free rectilinear trees), but there are good approximations available [14]. Still restricted to the rectilinear case, and given that the instances are typically relatively small in the case of analog chip design, one can also consider computing a truly minimal Steiner tree, using, e.g., GeoSteiner [19, 20]. For more information, the paper by Hentschke et al. [11] is a good survey on exact results and approximations for rectilinear Steiner trees, taking into account different priorities of optimization.

4 Integer Programming and Approximations

In this section we propose a mathematically more rigorous approach, which extends our heuristic from Section 2 but was too elaborate to incorporate in the prototype tool during the short time span of the study week.

This is a column generation approach (which can also be found in [10] in somewhat similar form), an approach that has successfully been applied to different real-world optimization problems (see [5]). While we give all formulations in terms of paths between pairs of pins, they work (with one limitation) also for larger nets.

4.1 Column Generation

Let us start with decomposing the problem into two levels. The top (or master) level is the following: given all nets and for each a pool of paths connecting them (possibly all such paths), we want to select one path for each net, such that the resulting wiring satisfies our hard constraints and is of high quality with respect to the optimization goals.

This is an integer linear program (see precise formulation below), which is known to be computationally hard to solve to optimality [13]. And while the analog design is not excessively large, here we get a huge number of variables: one for each pair of a net and a possible path for this net.

We note here that one can reduce the size of the graph involved by using less vertices and introducing different weights on the edges depending of the capacity of the space between the vertices. One could construct such a graph with a *Generalized Voronoi Diagram*. This method has been used for path planning in games (see, e.g., [8]), where characters have to move through a landscape and avoid obstacles. In chip design, electrons move through the wires and avoid components (except for the locations of their pins). We can define a Generalized Voronoi Diagram around the components. The edges in this diagram result in a collection of corridors going through the central areas of the open space between the components. In this way, they result in edges for our routing network. For each pin we add an edge representing the shortest line connecting that pin to the network. In general, this network will be smaller in terms of vertices and edges than the grid which is attractive from a computational point of view. Observe that in this setup multiple wires can go through an edge or vertex. One drawback of this is, however, that the solution is not yet a complete description of a physical layout. But one could first determine the corridor a net will use, and then solve sub-problems in this smaller grid. There are some more technicalities to be considered here, so we will just leave it as a suggestion for further considerations.

In any case, by far not all possible paths are of interest for us. In fact, most are unnecessarily long or could even include loops.

Thus, we restrict the master problem to a small pool of paths and introduce a second level, where we try to find good paths outside the pool (using the dual solution from the restricted problem) to improve the routing.

4.2 Formulation

Let (V, E) be the underlying network, i.e., the grid of Section 2 or an alternative network. We label the nets by $1, \dots, m$, and denote with \mathcal{P}_i the set of all possible paths for net i .

Define further $l_{ip} = \sum_{e \in p} l_e$ as the length of path $p \in \mathcal{P}_i$ (the lengths l_e are the weights we give the edges, depending on the optimization goals, e.g. according to Algorithm 13 in Section 2.3), let

$$a_{ep} = \begin{cases} 1 & \text{if edge } e \text{ is in path } p; \\ 0 & \text{otherwise,} \end{cases}$$

and define a_{vp} in the same fashion for the vertices that are not in one of the nets. Finally, we define the edge capacity c_e^{edge} as the maximum number of wires routed over edge e , and similarly c_v^{vertex} as the maximum number of wires that are allowed to cross through vertex v . For the grid used in Section 2.3, all values c_e^{edge} and c_v^{vertex} are set to 1.

Our variables are $x_{ip} \in \{0, 1\}$ where $x_{ip} = 1$ indicates that path $p \in \mathcal{P}_i$ is selected for net i . Then our master Integer Linear Program (master ILP) is

$$\begin{aligned} \min \quad & \sum_{i=1}^m \sum_{p \in \mathcal{P}_i} l_{ip} x_{ip} \\ \text{s.t.} \quad & \sum_{p \in \mathcal{P}_i} x_{ip} = 1 \quad (i = 1, \dots, m), \end{aligned} \quad (1)$$

$$\sum_{i=1}^m \sum_{p \in \mathcal{P}_i} a_{ep} x_{ip} \leq c_e^{\text{edge}} \quad \forall e, \quad (2)$$

$$\sum_{i=1}^m \sum_{p \in \mathcal{P}_i} a_{vp} x_{ip} \leq c_v^{\text{vertex}} \quad \forall v \text{ not in a net}, \quad (3)$$

$$x_{ip} \in \{0, 1\}. \quad (4)$$

Constraints (1) ensure that exactly one path is selected for every net. Constraints (2) and (3) ensure that the edge and vertex capacities are respected.

To deal with the large number of variables, we are going to solve the problem by column generation. We start with a limited subset of the variables and solve the LP-relaxation (i.e., $x_{ip} \geq 0$) for this subset only. For example, we could use the solutions from Section 2. This way we obtain the restricted master LP. Then we solve the pricing problem, i.e., we look for variables that are not yet included in the restricted master LP and can improve the solution.

If such variables are found, they are added to the restricted master LP, it is solved again, after which pricing is performed, and so on. If pricing does not find any new variables anymore, we know that the master LP has been solved to optimality.

Unfortunately, this solution is not likely to be an integer solution. We discuss methods for finding an integer solution in Section 4.4.

4.3 The Pricing Problem

From the theory of linear programming it is well-known that for a minimization problem increasing the value of a variable will improve the current solution if and only if its reduced cost is negative. The pricing problem then boils down to finding the variable with minimum reduced cost.

Let λ_i , π_e , and σ_v be the dual variables of the net, edge capacity, and vertex capacity constraints, respectively. Now the reduced cost of x_{ip} is given by

$$l_{ip} - \lambda_i - \sum_e a_{ep} \pi_e - \sum_v a_{vp} \sigma_v.$$

We are going to solve the pricing problems for each net separately. Note that a_{ep} and a_{vp} are the decision variables and that they have to form a path connecting the net. Clearly, the values a_{vp} (the vertices on the path) are determined by the values a_{ep} (the edges on the path). For each vertex v on the path we have cost $-\sigma_v$. Since on a path each vertex has degree 2 (except for the first and the last one, but these are in a net), we can remove the variables a_{vp} from the pricing problem by adding cost $-\frac{1}{2}\sigma_v$ to each edge adjacent to v . The reduced cost is now given by

$$\sum_e a_{ep} \left(l_e - \pi_e - \sum_{v \in e} \frac{1}{2} \sigma_v \right) - \lambda_i.$$

The pricing problem for net i thus reduces to finding a shortest path, where the edge lengths are modified by the dual variables.

From the theory of linear programming we know that $\pi_e \leq 0$ and $\sigma_v \leq 0$. Therefore, the cost of the edges are non-negative and the pricing problem can be solved by Dijkstra's algorithm.

For a net i with more than two pins, the \mathcal{P}_i are all possible Steiner trees, and hence at this point of the algorithm we are looking for a Steiner minimal tree. To save CPU-time, we can approximate the minimal Steiner tree, and only determine the optimal Steiner tree in case the approximation does not find a solution with negative reduced cost. If we decide to only approximate, we still have a high probability to find a very good solution to the LP-relaxation.

4.4 Integer Solutions

As we mentioned in the introduction, this approach also gives us a measure of the quality of the routing solution, because the solution of the LP-relaxation (which we solved to optimality) is a lower bound on the costs of the routing. To actually produce an integer solution, we can apply different strategies, which are only shortly mentioned here.

- We can perform branch-and-price, i.e., apply branching and proceeding with column generation (see, e.g., [3]). This will not lead us away from optimality.
- We can apply an ILP solver to the restricted master problem, which in all likelihood will have a manageable amount of variables.
- We can apply heuristics based on the LP solution. For example, we can first fix all paths that were selected with value 1. Then we proceed by selecting one by one paths with maximal fractional value that fit (in term of vertex and edge capacities) with the set of paths that were already selected. If we end with a solution with unconnected pins, we complete the solution using the heuristic from Section 2.

As was the case for the heuristic for the prototype tool, this method can also be applied if part of the routing is fixed, as this is nothing else than removing certain edges from our grid.

Comparing the ILP method of this section with the heuristic of Section 2, we note as advantages of ILP that it provides quality guarantees, it can be combined with the heuristic, and that it can be used on a smaller graph than the grid in the heuristic, which may save computation time. An advantage of the heuristic is that it is easy to implement, and that it often gives a fast and satisfactory solution. Summarizing, we see this ILP method as a natural extension of the heuristic, to be implemented if the heuristic gets too slow, if the solutions don't seem adequate, or to determine a quality measure of the heuristic.

5 Conclusion

Given the limited duration of the study week and the complexity of the problems connected to chip design, we decided to focus on one aspect which we felt could ease the work of the chip designers at NXP. Hence we tried to find an algorithm for connecting nets in a predefined layout, which is as flexible and customizable as possible, facilitating the designer to choose priorities between the different aspects that should be optimized, which is stable under local changes (if needed), and which gives reliably the same answers for identical inputs. We were able to present our algorithm in the form of a prototype tool (see Section 2.3) which showcases all these aspects. Furthermore, we describe a more generalized approach which provides a quality measure of the solution and improves our strategy to deal with larger inputs (see Section 4.1).

The study week permitted us to get acquainted with a large branch of new and interesting mathematics, as well as provide NXP with a useful prototype solution (Figure 4) for their routing problem.

References

- [1] C.J. Alpert, T.C. Hu, J.H. Huang, A.B. Kahng, and D. Karger. Prim-Dijkstra tradeoffs for improved performance-driven routing tree design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(7):890–896, 1995.
- [2] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, pages 2–11, 1996.
- [3] Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1996.
- [4] Cid Carvalho De Souza and Celso Carneiro Ribeiro. Heuristics for the minimum rectilinear Steiner tree problem: new algorithms and a computational study. *Discrete Applied Mathematics*, 45(3):205–220, 1993.

NXP participates in challenge event with Dutch universities

The "Studygroup Mathematics with Industry" is a yearly event where academic mathematicians work for a week on problems presented by the industry. At this year's event, from January 24–28 at Vrije Universiteit Amsterdam, NXP participated by presenting a complex challenge: the automatic placement and routing in analog layouts. A team of eight PhD students and academic researchers presented a well-working prototype simulation tool by the end of the week. This event is one of many activities with universities that NXP supports and actively participates in.

A team of eight enthusiastic PhD students and academic researchers from TU Delft, Utrecht University, CWI and Groningen University worked on this problem, receiving input and feedback from Joost Rommes and Theo Beelen from NXP, and presented their results on Friday January 28. Besides valuable literature references, also a well-working prototype simulation tool was developed, based on advanced graph and optimization algorithms. This tool allows users to set real-life layout requirements (wire distance and spacing, minimize number of used layers, etc) and see results in 3D on the fly! Last but not least, their unbiased analysis of the problem lead to new insights that NXP can use to enhance the methodology. Apart from the technical advances on the presented problems, the event was a fine opportunity to establish contacts between industry and academia, enabling to get acquainted with each other, and leading to challenging new research areas.



Figure 1: Bas Fagginger Auer (UU) explains the proposed solution for the routing problem of NXP

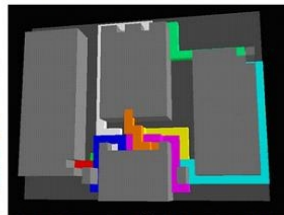


Figure 2: Impression of the interactive routing tool developed for NXP

The event was financially supported by University Relations of NXP.

For more information, please contact joost.rommes@nxp.com and theo.g.l.beelen@nxp.com

Figure 4: The work done during the study week was well received by NXP and was mentioned in their newsletter shortly after the final presentation.

- [5] Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, editors. *Column generation*, volume 5 of *GERAD 25th Anniversary Series*. Springer, New York, 2005.
- [6] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1:269–271, 1959.
- [7] M. R. Garey and D. S. Johnson. The Rectilinear Steiner Tree Problem is NP-Complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.
- [8] R. Geraerts. Planning short paths with clearance using explicit corridors. In *IEEE International Conference on Robotics and Automation*, pages 1997–2004, 2010.
- [9] P. E. Hart, N. J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
- [10] S. Held, B. Korte, D. Rautenbach, and J. Vygen. Combinatorial Optimization in VLSI design. In V. Chvátal and N. Sbihi, editors, *Combinatorial Optimization: Methods and Applications*. IOS Press, to appear.
- [11] Renato F. Hentschke, Jaganathan Narasimham, Marcelo O. Johann, and Ricardo L. Reis. Maze routing Steiner trees with effective critical sink optimization. In *Proceedings of the 2007 international symposium on Physical design, ISPD '07*, pages 135–142, New York, NY, USA, 2007. ACM.
- [12] Huibo Hou, Jiang Hu, and S.S. Sapatnekar. Non-Hanan routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(4):436–444, April 1999.

- [13] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Plenum Press, New York, 1972.
- [14] Christine R. Leverenz and Mirosław Truszczyński. The rectilinear Steiner tree problem: algorithms and examples using permutations of the terminal set. In *Proceedings of the 37th annual Southeast regional conference (CD-ROM)*, ACM-SE 37, New York, NY, USA, 1999. ACM.
- [15] Chih-Hung Liu, Shih-Yi Yuan, Sy-Yen Kuo, and Szu-Chi Wang. High-performance obstacle-avoiding rectilinear Steiner tree construction. *ACM Transactions on Design Automation of Electronic Systems*, 14:45:1–45:29, 2009.
- [16] Dirk Müller, Klaus Radke, and Jens Vygen. Faster min–max resource sharing in theory and practice. *Mathematical Programming Computation*, 3:1–35, 2011. 10.1007/s12532-011-0023-y.
- [17] S. Peyer, D. Rautenbach, and J. Vygen. A generalization of Dijkstra’s shortest path algorithm with applications to VLSI routing. *Journal of Discrete Algorithms*, 7:377–390, 2009.
- [18] J. Scott Provan. An approximation scheme for finding Steiner trees with obstacles. *SIAM J. Comput.*, 17(5):920–934, 1988.
- [19] David Warme, Paweł Winter, and Martin Zachariasen. GeoSteiner [Computer Software]. www.diku.dk/hjemmesider/ansatte/martinz/geosteiner/. (version 3.1).
- [20] Martin Zachariasen. Rectilinear full Steiner tree generation. *Networks*, 33(2):125–143, 1999.
- [21] Hai Zhou. Efficient Steiner tree construction based on spanning graphs. In *Proceedings of the 2003 international symposium on Physical design, ISPD ’03*, pages 152–157, New York, NY, USA, 2003. ACM.

Statistical Modelling of Pre-Impact Velocities in Car Crashes

*Wouter Kager** *Ivan Kryven[†]* *Keith W. Myerscough[‡]*
Timo van Opstal[§] *Thomas Rot**

Abstract

The law wants to determine if any party involved in a car crash is guilty. The Dutch court invokes the expertise of the Netherlands Forensic Institute (NFI) to answer this question. We discuss the present method of the NFI to determine probabilities on pre-impact car velocities, given the evidence from the crash scene. A disadvantage of this method is that it requires a prior distribution on the velocities of the cars involved in the crash. We suggest a different approach, that of statistical significance testing, which can be carried out without a prior. We explain this method, and apply it to a toy model. Finally, a sensitivity analysis is performed on a simple two-car collision model.

1 Introduction

Car crashes are unfortunately still common occurrences on the busy roads of the Netherlands. While it is of course preferable to prevent crashes from happening, as long as they do occur, it is important to determine if any party involved can be considered *guilty*. To determine this, Dutch courts invoke the expertise of the Netherlands Forensic Institute (NFI). The most common question asked in these situations is whether either party was driving too fast.

The NFI must answer this question based on data acquired at the crash scene by the relevant authorities. This is the *evidence*. To determine the velocities of both vehicles before the crash based on this evidence, two problems must be faced. Firstly, the physics of a vehicle collision has to be modelled accurately. There exists software which does this, but at the moment only forwards in time (see section 1.2). Secondly, the evidence is usually incomplete and has limited accuracy, as is discussed in section 1.1. The NFI tackles these issues with a Monte Carlo approach, briefly described in section 2, using the software PC-Crash. In this way an attempt is made to estimate the conditional distribution of the pre-crash velocity of a vehicle, given the evidence.

We argue that estimating this distribution requires a prior distribution on the pre-crash velocity, cf. section 3. The choice of prior is open to debate. The NFI assumes

*VU University

[†]University of Amsterdam

[‡]Centrum voor Wiskunde en Informatica

[§]TU Eindhoven

a uniform prior, but explains to the court that their calculations can be repeated with a different prior, if desired. We present an alternative approach, that of statistical significance testing, in section 4, which can be carried out without assuming a prior. This approach associates p -values to hypotheses, but the interpretation of these values is not a trivial matter (see section 4.2), and is also dependent on prior information. On the positive side, both the NFI and Dutch courts of law have experience in using p -values. Section 5 studies an example of the procedure applied to a toy problem.

Finally, a separate sensitivity analysis is presented in section 6. This analysis is performed on a simple two-car collision model with energy dissipation. Despite the simplicity of this model, it gives valuable insight into the sensitivity of the pre-crash velocity to the post-crash data.

1.1 Evidence Found at the Crash Scene

Whenever a car crash occurs, local authorities are called in. If the crash did not occur in a place extremely important for traffic flow (on a highway, for example), the commanding officer collects data, e.g. takes pictures of the cars, measures their positions and marks any tire tracks. It is important to note that the NFI is never present on a crash scene. The amount and accuracy of data is dependent on the diligence and experience of the commanding officer.

This means that collected data for each crash is different. Not only human behaviour is a factor in this. Natural surroundings also play a major role. For instance, if the crash occurs next to a lake, one of the cars may end up in the lake giving a huge amount of uncertainty in the end position of the car. All in all, we remark that the evidence depends on the situation and can have large uncertainties in the different parameters. It is therefore essential to have a sound and lucid method for dealing with these uncertainties. The method of statistical significance testing presented in section 4 satisfies these criteria.

1.2 The Software PC-Crash

PC-Crash is a deterministic simulation model for car crashes. The model consists of two parts: a collision model and a propagation model. The user sets up a scenario, placing cars and entering parameters. The program runs the propagation model until the cars make impact. An instantaneous collision is calculated, after which the propagation model is used to calculate the post-impact trajectories. In this paper we do not consider the correctness of this simulation model. We assume that it is perfect. Other studies [5] have shown that the model is reasonably correct for uncomplicated collisions. By uncomplicated we mean that we do not have factors which are not modelled in PC-Crash. For example, PC-Crash does not have the possibility of modelling cars that fall apart. It is also known that speed bumps are poorly modelled in the software. In those cases we cannot use PC-Crash.

The procedure proposed below is not dependent on the specific simulation program that is used. If a better program is developed we can incorporate this program into the procedure without any problem.

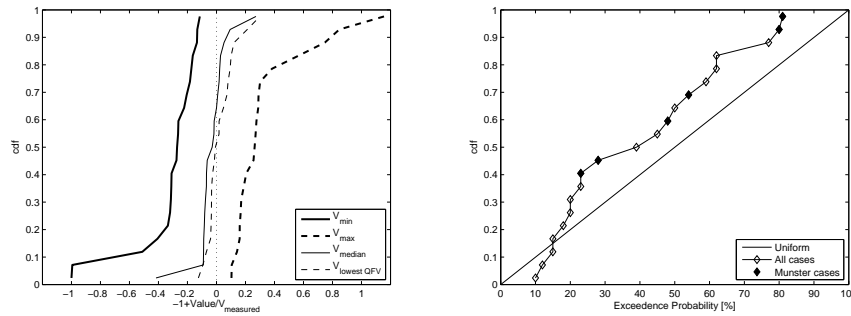


Figure 1: Left: the empirical cumulative distribution functions (cdf's) of the relative differences between the measured velocities and different ensemble velocities produced by MC-Crash. Right: the empirical cdf of the exceedance probabilities produced by MC-Crash.

2 Monte Carlo Simulation by the NFI

The approach developed by the NFI is a Monte Carlo procedure called MC-Crash, which is presented in [1]. An even more detailed set of instructions for researchers is given in [2]. [3] present some further information on the topic, but this does not represent the current methodology of the NFI. In summary, the NFI approach relies on random sampling from the input parameters (or “model parameters”) and measurement errors, and running PC-Crash for each parameter set. A quality function Q is used to determine how well a PC-Crash result matches the evidence. Private correspondence with Aart Spek, our contact at NFI, has taught us that the simulations are ended when the distribution of the 500 best parameter sets is visually close to that of the best 250, and that this usually requires 0.5–2 million samples. The PC-Crash output of the best 500 samples are stored.

In the NFI procedure, PC-Crash is treated as a black box function, which is assumed to model the physics involved correctly (cf. section 1.2). This has been verified by TNO at the request of the NFI [4, 5]. A validation of the Monte Carlo procedure of the NFI is presented in [4]. We now give a brief summary of this validation report.

The validation relies solely on comparing MC-Crash results to controlled test crashes. In these tests the pre-impact velocity was measured at the scene, making it possible to compare MC-Crash results with reality. As the output of MC-Crash is an ensemble of PC-Crash outputs, various statistical properties of this ensemble can be studied. For each test case, the velocity measured at the scene is compared with the minimum, maximum and median velocity of the MC-Crash ensemble, and also with the velocity of the sample which has the lowest Q . Finally, the measured velocity is compared with the so-called exceedance probability of the ensemble, which is the percentage of samples in the ensemble where the PC-Crash impact velocity exceeds the measured value.

The comparison between the MC-Crash results and the measured velocities can be visualised in two graphs (figures 4 and 5 in [4]), which we have reproduced in figure 1. In the validation report it is stated that “the exceedance probabilities ... [are],

ideally, uniformly distributed between 0 and 100 percent”. Figure 1 gives us no reason to doubt that this is the case. As we can also see in this figure, for all test cases the measured impact velocities are between the minimum and maximum velocities produced by MC-Crash. Spek concludes that the median velocity and the velocity of the sample with the lowest Q “can be considered as estimators of the impact velocity”.

We remark that it is far from trivial how the choice of the quality function Q and the stopping criterion employed by the NFI influence the statistical properties of the final ensemble. Time restrictions prevent us from investigating these issues further. However, figure 1 does suggest that Monte Carlo simulation using PC-Crash can indeed give useful information about impact velocities from the crash scene evidence. In the following sections, we will explain what kind of conclusions we believe can (and cannot) be drawn from the evidence using simulation software such as PC-Crash without a prior distribution on the pre-crash velocities.

3 The Model

The software PC-Crash converts an initial state to post-impact trajectories, cf. section 1.2, and the evidence at the crash scene is a subset of this space. This is abstracted to the following diagram:

$$X \xrightarrow{\text{PC-Crash}} Y \xrightarrow{\text{filter}} Z . \quad (1)$$

The map PC-Crash is the simulation program. The space X is the space of input parameters and Y is the space of output parameters of PC-Crash. The latter space is extremely big, as it contains the complete post-crash trajectories of the cars involved. The space Z describes all measurable data at the crash scene. This data is incomplete, as was discussed in section 1.1. We model this by applying a filter on the output space Y . This filter retains only a small part of the information provided by the PC-Crash software, for instance only the end positions and orientations of the cars. It is important to note that Z will be chosen differently for different crashes.

We are not interested in what happens in the space Y , because we only have access to the data retained in Z from the crash scene. Therefore, we simplify the model once more. We denote the composition of PC-Crash and filter by $f = \text{filter} \circ \text{PC-Crash}$. What remains is the diagram

$$X \xrightarrow{f} Z . \quad (2)$$

The evidence e obtained from the crash is a point in Z , complemented with information about the “measuring error” in each of the measured components of e . This information can be used to construct a model for the measurement performed at the crash scene. If the actual input parameters for the crash would have been $x \in X$, then the outcome at the crash scene would have been the point $z = f(x) \in Z$. Under the assumption that x describes what actually happened, we then model the outcome of the measurement at the crash scene as the point $z + S$, where S is a random variable describing the uncertainty in the measurement. We assume that this measurement error is drawn from a distribution μ , which does not depend on z and is centred around 0. The conditional distribution of the measurement $z + S$, given that the outcome of the crash is z , is then obtained by shifting μ to the centre z . We denote it by μ_z . It is up

to the experts to decide which distribution describes the measurement error best. In a typical situation (think of measuring the end position of a car), it might be given by independent Gaussians in each coordinate.

We stress here that we do not have enough information to completely model the measurement at the crash scene: we only know its *conditional* distribution *given* that the crash had a particular outcome. As a consequence of this, we can make probabilistic statements about the evidence found at the crash scene, *under the assumption* that a particular pre-crash (input) scenario occurred. In the next section, we will discuss in more detail how to do this. However, we would like to turn things around and draw conclusions about the likelihood of different pre-crash scenarios given the evidence. It is important to realise that this will be impossible without *prior* information on the likelihood of different input scenarios (or a prior distribution on the space X). We will return to this issue in section 4.2.

4 Statistical Significance Testing

In a car crash involving two cars, denoted A and B , we might typically be interested in the following type of hypotheses:

1. A was speeding, but B was not;
2. B was speeding, but A was not;
3. A and B were both speeding;
4. Neither A nor B was speeding.

We will consider such hypotheses in the classical framework of statistical hypothesis testing. The idea is that we compute the probability of the measurement at the crash scene being at least as extreme as what we have actually observed, under the assumption that our hypothesis is correct. This probability is called the p -value. A very small p -value could give rise to the rejection of the hypothesis, since under the assumption that the hypothesis is correct, one would see the observed data only with a very small probability. Often, one takes as “null-hypothesis” the negation of the hypothesis one wants to show to hold true. If its p -value is sufficiently small, one rejects this negation, and this is evidence supporting the hypothesis one is interested in.

4.1 The Formalism

Suppose that we want to test a certain hypothesis $H_0 \subset X$, in the setting of the model described in section 3. Here one could think for instance of an H_0 containing all points in X where the initial speed of car A is smaller than some given speed v . The idea is now that we create a subset C of Z , called the *critical region*, such that finding the evidence in C can be considered unlikely under the hypothesis H_0 . It is of crucial importance that we define C independently of the actual outcome of the performed measurement, otherwise we would always be able to arrange things such that we could draw the conclusion we were after. So, to construct the critical region we have to use our model of the measurement errors, but we are not allowed to use the evidence e .

In order to define C we use an observable quantity, called a *test statistic*. For simplicity, we describe the procedure here for the case where the measurement errors are modelled assuming independent Gaussian errors in all coordinates, but the principle can be extended to cases where the measurement errors are modelled in a different way (although it might be more debatable what is the correct test statistic to use then). So suppose that $Z = \mathbb{R}^n$ and that μ is a normal distribution centred at 0, with standard deviation σ_i in the i -th coordinate, so that $\mu_{f(x)}$ becomes a normal distribution with the same standard deviations in all coordinates, centred at $f(x)$. Then, under the assumptions that the initial state was x , a proper choice for the test statistic would be

$$T_x(z) = \sum_{i=1}^n \frac{(z_i - f_i(x))^2}{\sigma_i^2}. \quad (3)$$

This choice is motivated by the fact that the level sets of the density of $\mu_{f(x)}$ correspond to the level sets of T_x .

Now we can define the critical region on level δ :

Definition 1. Fix $\delta > 0$. Define

$$C_x^\delta = \{z \in Z : T_x(z) \geq \delta\}. \quad (4)$$

The critical region on level δ for the hypothesis H_0 is the set

$$C^\delta = \bigcap_{x \in H_0} C_x^\delta = \{z \in Z : T(z) \geq \delta\}, \quad (5)$$

where $T(z) = \inf_{x \in H_0} T_x(z)$ is the test statistic under the hypothesis H_0 .

The interpretation of these critical regions is fairly straightforward. The test statistic T_x measures the “distance” of the measurement to a proposed outcome $f(x)$ of the crash. If the measurement is too far away from this proposed outcome, i.e. if T_x is too large, then we do not believe that the actual outcome of the crash could have been $f(x)$, hence that the actual input parameters could have been x . A complication arises when we want to make a statement about a composite hypothesis H_0 consisting of multiple points in the input space X . In these cases, to reject H_0 , we require that the measurement is far from $f(x)$, no matter which $x \in H_0$ we consider. Hence the intersection over $x \in H_0$ in (5). In terms of the test statistic T , the critical region C^δ is precisely the region where the test statistic exceeds the value δ , since $T(z)$ is the minimal value of $T_x(z)$ attained for any $x \in H_0$.

If we have evidence found at the crash scene, that is, a point $e \in Z$, then we can search for the smallest δ such that $e \in C^\delta$:

Definition 2. The critical δ^* for H_0 is defined by

$$\delta^* = \inf\{\delta > 0 : e \in C^\delta\} = T(e), \quad (6)$$

and the p -value of H_0 is the maximal conditional probability that the outcome of the measurement would lie in C^{δ^*} , under the condition that the initial state was any point $x \in H_0$. That is, formally,

$$p = \sup_{x \in H_0} \mu_{f(x)}(C^{\delta^*}). \quad (7)$$

In practice, p -values—being an upper bound—are only useful when they are small, i.e. when e is far from $f(H_0)$. The above procedure, however, can be followed for any null-hypothesis H_0 . It will assign a (useless) p -value of 1 to H_0 if $e \in f(H_0)$ (in this case $\delta^* = 0$, thus $C^{\delta^*} = Z$ from which it follows that $p = 1$).

4.2 Interpretation

In words, the p -value is the maximal “probability of exceeding the evidence” among the elements in H_0 . A small p -value suggests that H_0 is not very likely, because it means that the evidence found at the crash scene is far from what you would expect to have measured under any possible pre-crash situation contained in H_0 . More precisely, conditioned on any point $x \in H_0$ being the pre-crash state, the probability of the measurement being at least as far from $f(x)$ as the evidence found is less than the p -value.

Note that we have not phrased our conclusions in terms of the probabilities of the various hypotheses. To do so would require prior knowledge which will—we believe—typically not be available or disputable. This has consequences when one wants to compare different hypotheses. Suppose, for example, that the hypothesis H_0 has a much smaller p -value than the hypothesis H_1 . This gives us more reason to reject H_0 than to reject H_1 . However, if the prior probability of H_0 is much higher than the prior probability of H_1 , it might well be the case that H_0 and H_1 are actually roughly equally likely given the evidence.

Likewise, we might consider the hypotheses H_v that a given car A was driving at a speed of at most v just before the crash, and compare the p -values of these hypotheses as a function of v . If these p -values stay very small for low v , and increase strongly around a specific speed v_0 , this can be seen as evidence supporting the presumption that the car was driving at a speed close to v_0 . However, without a prior on the pre-crash speeds, we cannot turn this into a probabilistic statement about the distribution of the car’s speed before the crash.

Indeed, the framework for comparing two hypotheses is different, and makes use of the so called likelihood-ratio: one wants to compare the likelihoods, or probabilities, of H_0 and H_1 given the evidence $e \in Z$. Assuming for a moment that both H_0 and H_1 consist of only one point (x and y , respectively), we can use the continuous version of Bayes’ rule and obtain

$$\frac{P(H_0|e)}{P(H_1|e)} = \frac{g_{f(x)}(e)}{g_{f(y)}(e)} \cdot \frac{P(H_0)}{P(H_1)},$$

where $g_{f(x)}$ is the density of the distribution $\mu_{f(x)}$, $g_{f(x)}(e)/g_{f(y)}(e)$ is the “likelihood ratio”¹, and $P(H_0)/P(H_1)$ the prior odds for H_0 against H_1 . These prior odds are not part of the model. To determine them, we need for instance to know something about the overall (prior) probability that a car is speeding. Given that we need the prior odds, it is clear that this does yield probabilities, but the interpretation is questionable unless one can remove doubts about the prior assumptions.

¹When H_0 and H_1 are composite, that is, consist of more than one point, the likelihood ratio is not immediately well-defined, but bounds can be obtained by considering maximal or minimal values of $g_{f(x)}(e)$. Especially when H_0 or H_1 is extended, so that $f(H_0)$ or $f(H_1)$ contain both points close to and far from e , this yields very impractical and useless numbers.

5 An Example of Significance Testing

To illustrate the hypothesis testing outlined in section 4, we consider a simple model problem. A car with mass m travelling with an absolute velocity $v > 0$ crashes into a wall and loses a certain amount of the energy E_d due to vehicle deformation. After the collision it has an absolute velocity $z \geq 0$, thus $Z = \mathbb{R}_{\geq 0}$. Knowing v and E_d , we are able to compute the post-collision speed z by a model $z = f(v, E_d)$. The function f is easily computed using conservation of energy (momentum is not conserved due to the external forces of the wall), yielding

$$f(v, E_d) = \sqrt{v^2 - 2E_d/m}. \quad (8)$$

Note that in this simple model we see the deformation energy as a parameter, i.e. an element of the space X . In reality this is not the case: material parameters along with the impact speed determine the dissipated energy. We should note that the space X is not \mathbb{R}^2 , but $X = \{(v, E_d) \in \mathbb{R}^2 : v^2 - 2E_d/m \geq 0\}$. Not all deformation energies and velocities can occur together, since the deformation energy must be less than or equal to the kinetic energy corresponding to the velocity at which the car was driving, i.e. the car cannot lose more energy in the collision than it had in the first place.

We now assume that the post-crash velocity e is measured with an error, where the error has the normal distribution with standard deviation σ and mean 0. We are interested in computing the p -value of the null-hypothesis $H_0 = \{v < \tilde{v}, E_d^a < E_d < E_d^b\}$. This hypothesis assumes that before the crash the car was travelling with a velocity less than \tilde{v} and that the deformation energy was between E_d^a and E_d^b . The hypothesis is non-empty if $\tilde{v}^2 - 2E_d^a/m \geq 0$, which we will assume in what follows.

We are able to compute the p -value using formula (7). Due to the simplicity of the model we can do this analytically. In general this is not possible, and numerical integrators and minimisers should be used instead. We start by computing δ^* :

$$\delta^* = \inf \{ \delta > 0 : e \in C^\delta \}. \quad (9)$$

This is equivalent to computing $T(e)$:

$$\begin{aligned} \delta^* = T(e) &= \inf_{(v, E_d) \in H_0} \frac{(e - f(v, E_d))^2}{\sigma^2} \\ &= \begin{cases} \frac{(e - f(\tilde{v}, E_d^a))^2}{\sigma^2} & \text{if } e > f(\tilde{v}, E_d^a) \\ 0 & \text{if } e \leq f(\tilde{v}, E_d^a) \end{cases}. \end{aligned} \quad (10)$$

The last computation used the fact that $f(H_0) = [0, f(\tilde{v}, E_d^a)]$, because f is increasing in v , and decreasing in E_d . If $e > f(\tilde{v}, E_d^a)$, the infimum is attained at $f(\tilde{v}, E_d^a)$. If $e \leq f(\tilde{v}, E_d^a)$, then there is a point $(v, E_d) \in H_0$ such that $f(v, E_d) = e$, thus $\delta^* = 0$. As discussed in section 4, it follows that $p = 1$ if $e \leq f(\tilde{v}, E_d^a)$. It remains to compute the p -value when $e > f(\tilde{v}, E_d^a)$. In this case, from a simple computation it follows that $C^{\delta^*} = \{z \in Z : z > e\}$. Thus we compute using equation (7),

$$\begin{aligned} p &= \sup_{(v, E_d) \in H_0} \mu_{f(v, E_d)}(C^{\delta^*}) \\ &= \sup_{(v, E_d) \in H_0} \frac{1}{\sqrt{2\pi}\sigma^2} \int_e^\infty \exp\left(-\frac{(z - f(v, E_d))^2}{2\sigma^2}\right) dz. \end{aligned} \quad (11)$$

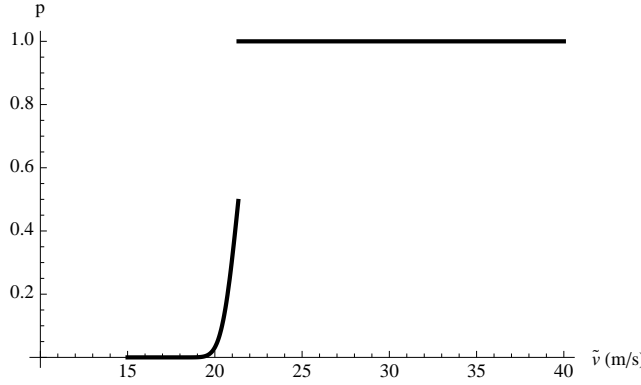


Figure 2: A plot of the p -values of the hypotheses $H_0 = \{v < \tilde{v}, E_d > E_d^a = 10^5 \text{ J}\}$. The measured post-crash velocity is $e = 16 \text{ m/s}$. The standard deviation is $\sigma = 1 \text{ m/s}$. The mass of the car is $m = 1000 \text{ kg}$. In other figures we show the response of the p -values to changes in the parameters σ , E_d^a , and the measured post-crash velocity e . The location of the sharp increase in the p -value suggests that the pre-crash speed was around this velocity.

The integral is easily expressed in terms of the error function:

$$\begin{aligned} p &= \sup_{(v, E_d) \in H_0} \frac{1}{2} \left(1 - \text{Erf} \left(\frac{e - f(v, E_d)}{\sqrt{2}\sigma} \right) \right) \\ &= \frac{1}{2} \left(1 - \text{Erf} \left(\frac{e - f(\tilde{v}, E_d^a)}{\sqrt{2}\sigma} \right) \right). \end{aligned} \quad (12)$$

In the last step we used that the function under the supremum is increasing in v and decreasing in E_d . We plug in the maximal v and minimal E_d in H_0 to obtain the result. By combining the previous computations, we conclude that

$$p = \begin{cases} \frac{1}{2} \left(1 - \text{Erf} \left(\frac{e - f(\tilde{v}, E_d^a)}{\sqrt{2}\sigma} \right) \right) & \text{if } e > f(\tilde{v}, E_d^a) \\ 1 & \text{if } e \leq f(\tilde{v}, E_d^a) \end{cases}. \quad (13)$$

We see that the p -value is insensitive to the upper bound on the deformation energy. This is due to the form of our null hypothesis. If we would compute the p -value for the null-hypothesis $H_0 = \{v > \tilde{v}, E_d^a < E_d < E_d^b\}$ we would find that the p -value is insensitive to the lower bound of the deformation energy.

In a realistic scenario it is impossible to compute the p -values analytically. We study some plots, obtained from the analytical result, that can be obtained in real scenarios with numerical computations using PC-Crash. In all plots we took the mass m of the car to be 1000 kg. We plotted the p -values of the hypothesis $H_0 = \{v < \tilde{v}, E_d > E_d^a\}$ against \tilde{v} in figure 2. The lower bound on the dissipated energy is $E_d^a = 10^5 \text{ J}$ and the standard deviation is $\sigma = 1 \text{ m/s}$. We see a sharp increase in the p -value just above $\tilde{v} = 20 \text{ m/s}$. At $\tilde{v} = 20 \text{ m/s}$ the p -value equals 3.2×10^{-2} , and at $\tilde{v} = 18 \text{ m/s}$ the p -value is 5.7×10^{-7} . As discussed in section 4.2, we therefore have a much larger confidence in rejecting the hypothesis that the car was driving at a velocity of at most 18 m/s than in rejecting the hypothesis that the car was driving at

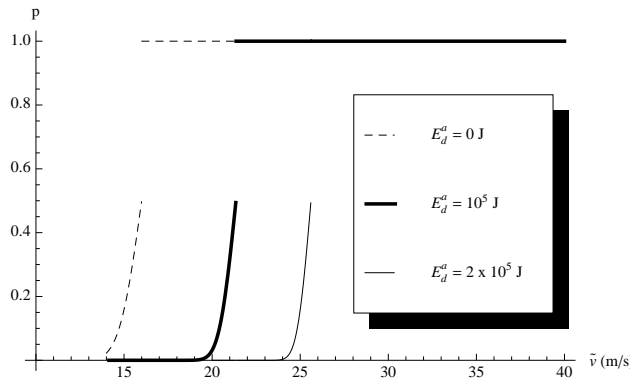


Figure 3: The plots depict the response of the p -values to varying dissipation energy. The p -values are plotted for different hypotheses $H_0 = \{v < \tilde{v}, E_d > E_d^a\}$ with varying E_d^a . The values of the other parameters are the same as in figure 2. We see that if we assume that more energy is dissipated, the increase in the p -values shifts (non-linearly cf. eq. (13)) to a higher pre-crash velocity.

a velocity of at most 20 m/s. However, without prior information we cannot state that, given the measured post-crash velocity, the probability of the car driving faster than 18 m/s is much larger than the probability of the car driving faster than 20 m/s.

Figure 3 shows the response of the p -values to changes in the value of E_d^a in the null-hypothesis. As expected, if we assume that more energy is dissipated, we are led to believe that the pre-crash velocity was higher. Figure 4 shows that an increasing uncertainty, i.e. a larger standard deviation, makes it harder to draw conclusions about pre-crash velocities. Finally, figure 5 investigates changes in the measured post-crash velocity. The behaviour is as expected: a higher post-crash velocity suggests that the car was driving faster before the crash. The discontinuities in all these plots occur at the smallest \tilde{v} such that $e \in f(H_0)$. From this point on the p -value equals 1, and becomes a trivial upper bound, hence is useless.

There is a small unmentioned subtlety in the previous computation. The post-crash velocity is a positive quantity and the probability distributions do have a finite probability mass for negative velocities. The assumption that the measurement error is normally distributed fails for small values of the velocity (theoretically, and practically as well). We have chosen to ignore this phenomenon, because the problematic values of the velocity (around $v = 0$ m/s) are many standard deviations away from the evidence.

6 Sensitivity-based Methods

This section contains a different approach to the problem. We consider a simple deterministic model for a collision of two cars. In this case we are able to invert the system of equations, i.e. we are able to compute the pre-crash velocities from the post-crash data. We are interested in understanding how small perturbations in the post-crash data affect the predicted pre-crash velocities. A powerful tool to analyse this are sen-

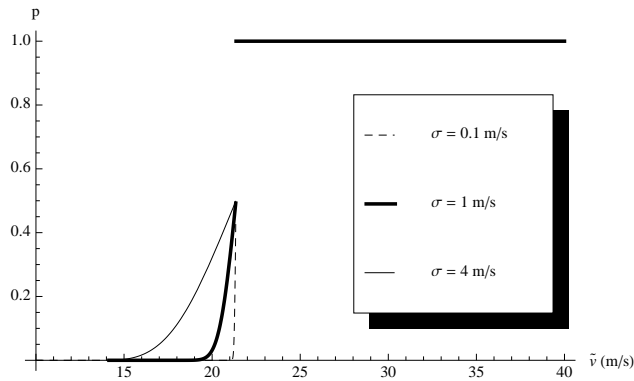


Figure 4: The plots depict the response of the p -values to a varying standard deviation σ . The p -values are plotted for different hypotheses $H_0 = \{v < \tilde{v}, E_d > E_d^a = 10^5 \text{ J}\}$, with varying standard deviation σ . Other parameters are the same as in figure 2. The increase in the p -values becomes less sharp with an increasing standard deviation, which makes it harder to draw strong conclusions.

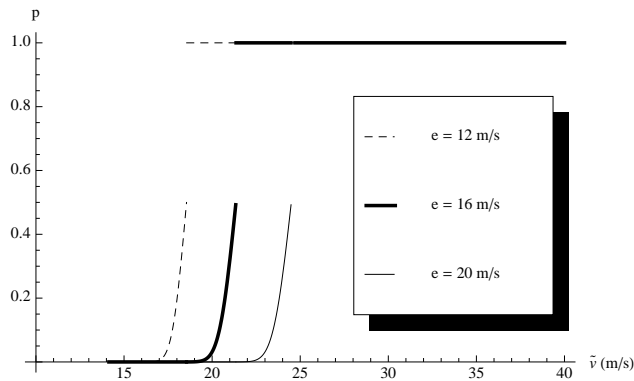


Figure 5: The plots depict the response of the p -values to a different measured post-crash speed e . The p -values are plotted for different hypotheses $H_0 = \{v < \tilde{v}, E_d > E_d^a = 10^5 \text{ J}\}$, with varying e . Other parameters are the same as in figure 2. If we measure a larger post-crash velocity, the increase in the p -values shifts to a higher pre-crash velocity, as expected.

sitivities. We will compute the sensitivities in this simple model and study them.

It should be noted that the sensitivities are ill-defined for collisions simulated using PC-Crash because the program is not invertible, i.e. we cannot uniquely reconstruct the pre-crash velocities from post-crash data. However, we do believe the discussion in this section gives valuable insight into the relative importance of measuring errors for the modelling of pre-crash velocities in real scenarios.

6.1 Deterministic Setting

We consider a simple collision model with two cars crashing into each other. We assume that the momentum exchange tangent to the collision is zero, so that the problem reduces to a one-dimensional collision. The cars obey the laws of momentum and energy conservation:

$$\eta u + \nu v = \eta u_+ + \nu v_+, \quad (14)$$

$$\frac{1}{2}\eta u^2 + \frac{1}{2}\nu v^2 = \frac{1}{2}\eta u_+^2 + \frac{1}{2}\nu v_+^2 + E_d. \quad (15)$$

In these equations (η, ν) , (u, v) and E_d denote vehicle masses, pre-impact velocities and impact dissipation energy, whilst (u_+, v_+) denotes the post-impact velocities.

This is a quadratic equation, and it is not difficult to compute the pre-crash velocities u, v , if we know the post-crash velocities u_+, v_+ , and the dissipated energy E_d :

$$u = (\nu v_+ + \eta u_+ \pm \nu \sqrt{D}) / (\nu + \eta) \quad (16a)$$

$$v = (\nu v_+ + \eta u_+ \mp \eta \sqrt{D}) / (\nu + \eta). \quad (16b)$$

The discriminant D is given by

$$D = (v_+ - u_+)^2 + 2 \frac{\eta + \nu}{\eta \nu} E_d. \quad (17)$$

The signs \pm depend on the data u_+ and v_+ . In general a one-dimensional collision model such as this one has two solutions. In one solution the colliding objects bounce off each other, and in the other solution they move through each other. Both collisions are governed by the same equations. An example of the latter type of solution is a bullet shooting through a tin can.

In our situation the cars do not move through each other. This means that if $v_+ > u_+$, we must have $u > v$, so we need the $+$ sign in equation (16a), and the $-$ sign in equation (16b). In the case $v_+ < u_+$ we must use the opposite signs. If $u_+ = v_+$, we cannot be sure which situation has occurred. This will cause a discontinuity at $u_+ = v_+ \neq 0$.

6.2 Sensitivities

For the remainder we analyse u . The situation for v is analogous. Small variations in the parameters u_+, v_+, E_d, η , and ν give rise to variations in u . This can be understood from the differential change in u :

$$du = \frac{\partial u}{\partial u_+} du_+ + \frac{\partial u}{\partial v_+} dv_+ + \frac{\partial u}{\partial E_d} dE_d + \frac{\partial u}{\partial \eta} d\eta + \frac{\partial u}{\partial \nu} d\nu. \quad (18)$$

We are interested in how these terms relatively influence du . We rescale all the terms and write

$$du = u_+ \frac{\partial u}{\partial u_+} \frac{du_+}{u_+} + v_+ \frac{\partial u}{\partial v_+} \frac{dv_+}{v_+} + E_d \frac{\partial u}{\partial E_d} \frac{dE_d}{E_d} + \eta \frac{\partial u}{\partial \eta} \frac{d\eta}{\eta} + \nu \frac{\partial u}{\partial \nu} \frac{d\nu}{\nu}. \quad (19)$$

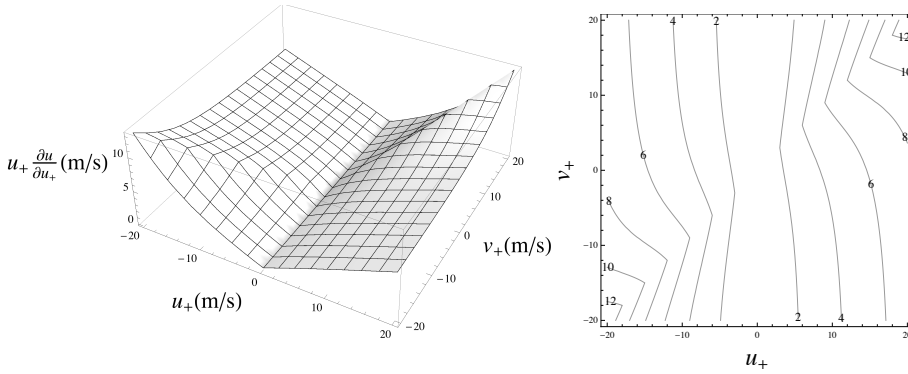


Figure 6: The sensitivity to u_+ is plotted, as a function of u_+ and v_+ . The other parameters were kept fixed: $\eta = 2000$ kg, $\nu = 1000$ kg, and $E_d = 5 \times 10^4$ J.

The equation expresses how u is modified by small relative changes in each of the parameters. Thus the absolute value of for instance $u_+ \frac{\partial u}{\partial u_+}$ determines the sensitivity of u to relative changes in u_+ . We call $u_+ \frac{\partial u}{\partial u_+}$ the sensitivity of u to u_+ . The sensitivities of u to u_+ , v_+ , and E_d are given by

$$\begin{aligned} u_+ \frac{\partial u}{\partial u_+} &= \left(\frac{\eta}{\eta + \nu} \pm \frac{\nu(u_+ - v_+)}{\sqrt{D}(\eta + \nu)} \right) u_+ \\ v_+ \frac{\partial u}{\partial v_+} &= \left(\frac{\eta}{\eta + \nu} \pm \frac{\nu(v_+ - u_+)}{\sqrt{D}(\eta + \nu)} \right) v_+ \\ E_d \frac{\partial u}{\partial E_d} &= \left(\pm \frac{1}{\eta \sqrt{D}} \right) E_d. \end{aligned}$$

In principle we could also have computed the sensitivities to η and ν , but we will focus on the sensitivities to u_+ and E_d . The sensitivity to u_+ is plotted in figure 6 as a function of u_+ and v_+ .

It is not difficult to obtain bounds for the sensitivities to the post-crash speed and dissipation energy from our explicit formulas. If we work under the assumption that $u_+^2 \geq v_+^2$, for instance, we can reason as follows. First we express the dissipation energy in terms of the post-crash kinetic energy of the first car by introducing a positive parameter α such that

$$E_d = \frac{4\alpha^2\nu}{\eta + \nu} \cdot \frac{1}{2}\eta u_+^2.$$

Since $(v_+ - u_+)^2 \leq 4u_+^2$, we can now bound the discriminant D by

$$4\alpha^2 u_+^2 \leq D \leq 4(1 + \alpha^2) u_+^2,$$

from which we obtain

$$\frac{\alpha^2}{\sqrt{1 + \alpha^2}} \frac{\nu}{\eta + \nu} |u_+| \leq \left| E_d \frac{\partial u}{\partial E_d} \right| \leq \alpha \frac{\nu}{\eta + \nu} |u_+|.$$

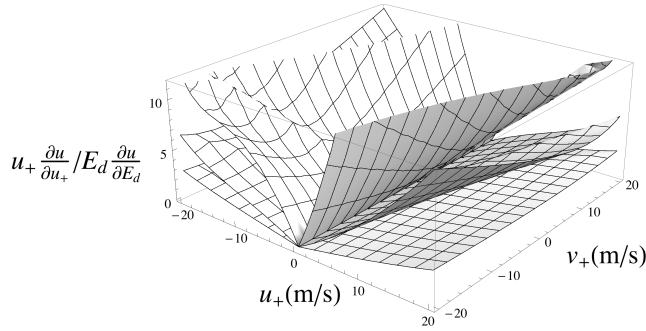


Figure 7: The ratio of sensitivities to u_+ and E_d is plotted, as a function of u_+ and v_+ , for different values of E_d . The top leaf is plotted for $E_d = 1 \times 10^4$ J. The middle leaf has a dissipation energy $E_d = 5 \times 10^4$ J, and the bottom leaf has $E_d = 2.5 \times 10^5$ J. For small E_d the sensitivity to u_+ is important, and for high E_d the sensitivity to E_d becomes more significant.

On the other hand, $(v_+ - u_+)^2 \leq 4u_+^2$ also implies that

$$D \geq (1 + \alpha^2)(v_+ - u_+)^2,$$

from which we conclude that

$$\frac{\eta - \nu/\sqrt{1 + \alpha^2}}{\eta + \nu} |u_+| \leq \left| u_+ \frac{\partial u}{\partial u_+} \right| \leq \frac{\eta}{\eta + \nu} |u_+|.$$

These bounds show that if the masses of the two vehicles are of the same order, the sensitivity to the dissipated energy will be larger than the sensitivity to the post-crash speed (of the first vehicle) when α is large. When α is small, on the other hand, the sensitivity to the dissipated energy stays small as well, and the sensitivity to the post-crash speed will be larger (provided the vehicle masses are not the same). It is to be noted here that large or small α correspond respectively to the dissipated energy being much larger or much smaller than the post-crash kinetic energy of the first vehicle. Thus, we can conclude that which sensitivity dominates depends on how the dissipated energy compares to the post-crash kinetic energy of the vehicles. This is visualised in figure 7.

6.3 Stochastic Analysis

In the simple toy model studied above, we could explicitly solve for the pre-crash speed u , given the output parameters (u_+, v_+, E_d) . If we are in such a situation, we may map the measured outcome of the crash directly to a pre-crash speed u , and we may wonder how good an estimate of the pre-crash speed this will give us. It is this question which we will address here.

To generalise the problem, we assume in this discussion that the output space Z is d -dimensional, and that every point $z = (z_1, \dots, z_d) \in Z$ maps uniquely to a pre-crash speed $u(z)$. The toy model discussed above fits in this picture if we take $d = 3$ and identify the coordinates (z_1, z_2, z_3) with the respective output parameters (u_+, v_+, E_d) . As in section 3, if we now assume that the actual outcome of the crash

is given by the point $z \in Z$, we can model a measurement of the outcome of the crash as the point $z + S$, where S is a random variable describing the measurement error. As before, we assume that this error is drawn from a distribution which is centred around 0.

To simplify the discussion, we furthermore assume the following:

1. The coordinates of S are independent. In this case the probability density function g of S is simply the product of the densities of its components. This is a simplification for presentation purposes, but not an essential condition for the method presented here.
2. The density of each coordinate of S is symmetric around 0, and the i -th coordinate has standard deviation σ_i .
3. The pre-crash speed depends sufficiently smoothly on the data around the given point z . For instance, in the example discussed above we require the relation (16a) to be sufficiently smooth in a neighbourhood of (u_+, v_+, E_d) we are interested in.

The 3rd-order Taylor series expansion of $u(z + s)$ around z is given by

$$u(z + s) = u(z) + \sum_{i=1}^d \frac{\partial u(z)}{\partial z_i} s_i + \frac{1}{2} \sum_{i=1}^d \sum_{j=1}^d \frac{\partial^2 u(z)}{\partial z_i \partial z_j} s_i s_j + O(|s|^3). \quad (20)$$

In the setup under discussion, each output point is associated in a deterministic way to a corresponding pre-crash speed. Thus we can map our measurement $z + S$ to the pre-crash speed $U = u(z + S)$. This new random variable describes which conclusion we would draw about the pre-crash speed from a measurement at the crash scene using our deterministic model, *under the condition* that z was the actual outcome of the crash (hence $u(z)$ was the actual pre-crash speed). If we know the map explicitly, we can now study properties of this random variable U , and determine how good an estimate of the actual pre-crash speed it will be. In particular, we can analyse how this estimate depends on the measurement errors in the different coordinates.

From the definitions of stochastic moments, for instance, the mean m_U and standard deviation σ_U of the random variable U are given by

$$m_U := \int_Z u(z + s) g(s) ds \approx u(z) + \frac{1}{2} \sum_{i=1}^d \left(z_i^2 \frac{\partial^2 u(z)}{\partial z_i^2} \right) \left(\frac{\sigma_i}{z_i} \right)^2; \quad (21a)$$

$$\sigma_U^2 := \int_Z (u(z + s) - \mu_U)^2 g(s) ds \approx \sum_{i=1}^d \left(z_i \frac{\partial u(z)}{\partial z_i} \right)^2 \left(\frac{\sigma_i}{z_i} \right)^2. \quad (21b)$$

We have expressed these moments in terms of the relative errors σ_i/z_i in each coordinate, and we then see the sensitivities to the different coordinates pop up in these expressions. The errors on these approximations are 4th order in the relative errors, because under the assumptions made above, the odd moments of the measurement errors vanish and different coordinates are independent. We see that the mean shifts according to the second derivatives of u , and that the standard deviation changes according to the squares of the sensitivities.

7 Conclusion

In this paper we propose an alternative approach for determining pre-crash velocities from crash scene evidence using an available black-box crash model. This *statistical significance testing* yields p -values for a set of pre-formulated statistical hypotheses. These are valuable indicators frequently used in modern statistical analysis and forensic science. It should be noted that there are no limitations on the complexity of the collision model. Any (third party) software is acceptable.

An example using a simple crash model was presented to help understanding the nature of uncertainty in the reconstructed data and how this is reflected by the p -values. This procedure can be adapted to use any other crash model, such as PC-Crash.

A sensitivity analysis was performed on a simple two-body collision. It was observed that sensitivity to energy dissipation depends on the amount of dissipated energy. It is particularly important to measure the dissipated energy with a high relative accuracy in collisions in which a large proportion of the kinetic energy is dissipated.

The presented approach provides a statistical foundation for a forensic expert analysing a car crash, and for presenting this data in court.

Acknowledgements

We would like to thank Geert Geeven, Ronald Meester, Thanh Son Nguyen, Bob Planqué, and Aart Spek for many fruitful discussions on this topic.

References

- [1] Reconstructie bots- en aanrijdsnelheden mbv MC-Crash. Technical Report 272501, NFI.
- [2] K.M. Hagendoorn. *Handleiding MC-Crash*.
- [3] A. Moser, H. Steffan, A. Spek, and W. Makkinga. Application of the monte carlo methods for stability analysis within the accident reconstruction software pc-crash. *Society of Automotive Engineers, Inc.*, January 2003.
- [4] A.C.E. Spek. Validatierapport MC-Crash. Technical report, NFI, January 2003.
- [5] H.H. Spit. Evaluation of PC-Crash - A Momentum-Based Accident Reconstruction Program. Technical Report 00.OR.BV.2712.1/HHS, TNO, 2000.